

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE Jan. 12, 1995		3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE Parallel Adaptive Finite Element: Software for Semiconductor Device Simulation			5. FUNDING NUMBERS	
6. AUTHOR(S) R.W. Dutton, K.H. Law, P.M. Pinsky, N. Aluru, B. Herndon				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Stanford University AEL 203 Stanford, CA 94305-4055			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) U. S. Army Research Office P. O. Box 12211 Research Triangle Park, NC 27709-2211			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The view, opinions and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other documentation.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.			12b. DISTRIBUTION CODE DTIC QUALITY INSPECTED	
13. ABSTRACT (Maximum 200 words) Parallel algorithms and fully functional application codes for 2D and 3D device analysis of semiconductor devices have been demonstrated. Advanced modeling based on a hydrodynamic formulation (HD) of the semiconductor transport equations and using a Galerkin Least Squares Finite Element Method (GLS-FEM) has demonstrated nearly ideal parallel performance for 2D MOS and Bipolar transistor applications across Intel (512 node, Delta) and IBM (16 node, SP/1) machines. Parallelization of conventional drift-diffusion (DD) based device solvers has broken new ground in both direct and iterative solvers. A well-known application code, PISCES, has been parallelized and ported across Intel, TMC, and IBM architectures with best results to date that now approach 6.5 GFlops sustained performance on a 128 node IBM SP/2. A prototype 3D code (STRIDE) which uses iterative methods has parallelized preconditioners for ILU(0), ILU(1), and ILUV and achieved excellent benchmarks on both the Intel and IBM machines. A 4.9 million grid problem run on the Intel Delta machine achieved 20% efficiency using 512 nodes and convergent solutions for a highly nonlinear bipolar transistor problem in 20 minutes per bias point. In support of both 2D and 3D TCAD applications, a new geometry-based structure generator called VIP3D was created. Quad- and oct-tree utilities were developed and used to support the gridding of complex IC structures benchmarked in this work. Results of industrial impact and collaborative interactions are also discussed.				
14. SUBJECT TERMS parallel computing, 3D semiconductor device simulation, finite-element analysis (FEM), parallel iterative solvers, parallel direct solvers, Galerkin Least Squares (GLS-FEM), hydrodynamic formulation, drift-diffusion, MOS transistor, bipolar transistor, adaptive gridding, quad-tree gridding, oct-tree gridding, solid geometry-based device modeling			15. NUMBER OF PAGES 17	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED			18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	
19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED			20. LIMITATION OF ABSTRACT UL	

19950216 055

Parallel Adaptive Finite Element Software for Semiconductor Device Simulation

Robert W. Dutton, Kincho H. Law, Peter M. Pinsky,
Narayana R. Aluru, Bruce P. Herndon

Stanford University
Stanford, CA 94305-4055

Abstract

Parallel algorithms and fully functional application codes for 2D and 3D device analysis of semiconductor devices have been demonstrated. Advanced modeling based on a hydrodynamic formulation (HD) of the semiconductor transport equations and using a Galerkin Least Squares Finite Element Method (GLS-FEM) has demonstrated nearly ideal parallel performance for 2D MOS and Bipolar transistor applications across Intel (512 node, Delta) and IBM (16 node, SP/1) machines. Parallelization of conventional drift-diffusion (DD) based device solvers has broken new ground in both direct and iterative solvers. A well-known application code, PISCES, has been parallelized and ported across Intel, TMC, and IBM architectures with best results to date that now approach 6.5 GFlops sustained performance on a 128 node IBM SP/2. A prototype 3D code (STRIDE) which uses iterative methods has parallelized preconditioners for ILU(0), ILU(1), and ILUV and achieved excellent benchmarks on both the Intel and IBM machines. A 4.9 million grid problem run on the Intel Delta machine achieved 20% efficiency using 512 nodes and convergent solutions for a highly nonlinear bipolar transistor problem in 20 minutes per bias point. In support of both 2D and 3D TCAD applications, a new geometry-based structure generator called VIP3D was created. Quad- and oct-tree utilities were developed and used to support the gridding of complex IC structures benchmarked in this work. Results of industrial impact and collaborative interactions are also discussed.

I. Introduction

The multi-dimensional analysis of semiconductor devices--both in two- and three-dimensions--provides the backbone of advanced technology development. Yet these analysis capabilities are computationally demanding and to date the infrastructure to support large scale 3D models has been insufficient to result widespread industrial use. At the same time, the scaling of IC devices and technology into the deep submicron regime in support of giga- and tera-flop applications increasingly demands such 3D technology computer-aided design (TCAD).

This contract is targeted at the development of advanced finite-element method (FEM) and other robust device analysis capabilities that can support parallel computational strategies to overcome analysis time and resource constraints. The leverage provided by powerful new parallel computers can thereby reduce the development time and costs for new advanced devices by orders of magnitude. Moreover, the advanced models and numerically stiff partial differential equations (PDE) used in this work serve as a key set of benchmarks for parallel computing that test the

machines and algorithms in the context of practical applications.

Hence, the objectives of this work focus on the complete parallelization of semiconductor device analysis codes and the benchmarking of their capabilities on state-of-the-art parallel computers. This work involves a range of supporting tasks and software technology. For example, new FEM technology as well as parallel element matrix assembly, nonlinear and linear solvers and supporting gridding technology all need to be developed and made functional in working applications. The approach taken in this work centers on three major areas of development:

1. Galerkin-Least-Squares finite-element methods have been used successfully for computational fluid dynamics. In this project, a similar formulation of the semiconductor equations is being developed. In addition, we are developing error estimators for the coupled system of elliptic Poisson and the hyperbolic advective-diffusive equations.
2. Both iterative and hybrid (direct/iterative) solution techniques are being pursued in order to enhance the robustness of simulations over the complete range of biasing and device configurations. This includes parallel preconditioners and iterative solvers for 3D problems; for 2D problems we are exploring robust iterative techniques.
3. Quad-tree/oct-tree-based gridding schemes are used to support the solver technologies and provide flexibility for adaptation and parallelization. Device geometry comes from a semiconductor wafer representation (SWR) and the methodology supports non-planar surfaces as well as dynamically changing geometry as a result of process simulation (see the SPRINT-CAD project for further details).

During the course of this work there have been major results achieved in all these areas. The parallel benchmarks achieved on three commercial machines have now clearly demonstrated the viability of 3D TCAD using parallel computers. In addition, there has been rapid and substantial growth in both industrial use of the hardware technology and vendors now appear ready to support commercialization. The following sections are organized to discuss each of the areas listed above. The body of the report gives an overview of the work and major benchmarks as they relate to parallel computing. The supporting details are covered in appendices where reprints and preprints of key publications are included for completeness. Finally, there is a section related to technology transition, both in terms of users and potential vendors.

Accession For	
NTIS	<input checked="checked" type="checkbox"/>
CRA&I	<input type="checkbox"/>
DTIC	<input type="checkbox"/>
TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification _____	
By _____	
Distribution / _____	
Availability Codes _____	
Dist	Avail and/or Special
A-1	

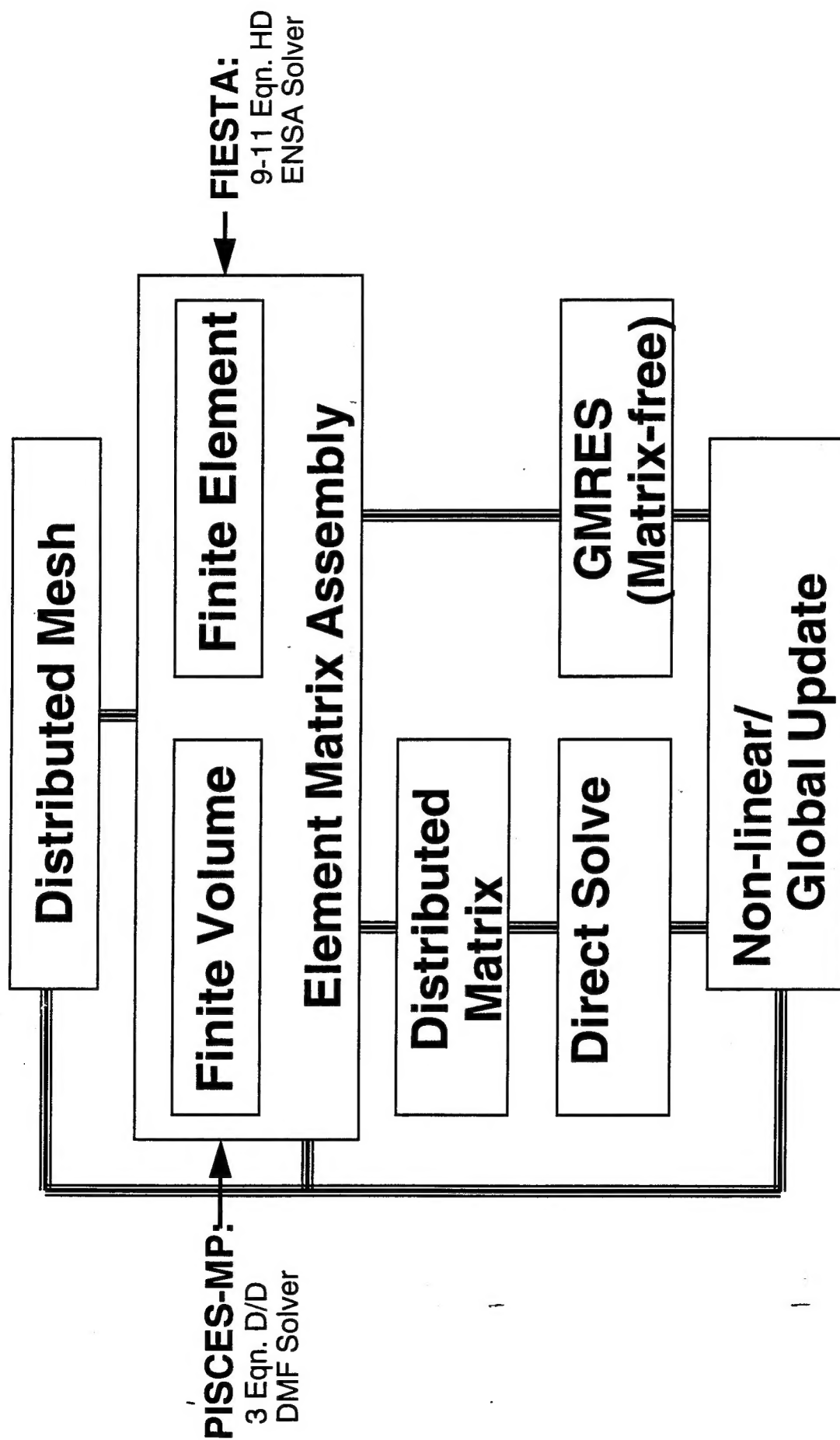


Figure 1: Schematic representation of parallelization of application codes for semiconductor device analysis--both the PISCES-MP and FIESTA codes are shown.

II. Overview of Parallel Device Analysis

The analysis of semiconductor devices involve the solution of both carrier transport and electrostatics that result from boundary conditions imposed on the device. The Boltzmann Transport Equation (BTE) is the most general representation of the carrier dynamics and can be solved by a variety of methods, depending on simplifications used. Here we focus on the use of assumptions regarding carrier statistics and integration over momentum space that reduces the general BTE to a set of PDEs. Specifically, the two classes of transport considered here are: 1) the hydrodynamic (HD) formulation which involves the conservation of carrier number, momentum, and energy (see Section III and Appendix A for more details), and 2) the drift-diffusion (DD) formulation where only the first moment of the BTE is used. The number of PDEs involved in each case is quite different; for the HD formulation, there are different levels of model complexity as outlined in Appendix A. The key point for discussion in the context of this project is the fact that over a range of models and number of supporting PDE systems for semiconductor device analysis, parallel solvers and complete applications have been demonstrated.

A schematic view of the architecture for creating parallel device analysis capabilities is shown in Figure 1. Both the HD and DD formulations used in this work are indicated along with the number of PDEs being solved in each case. In an effort to explore and understand the broadest needs and requirements for TCAD applications, both the FEM and a more traditional finite volume approach were considered. In the later case, Stanford has developed the well-known PISCES code for two-carrier analysis (holes and electron) based on the DD formulation. PISCES was later adapted for execution on message-passing distributed-memory parallel computers. This adaptation was modeled upon the earlier work of Lucas [1][2].

The HD analysis capabilities shown in Figure 1 are built on both a new formulation of the semiconductor problem using Galerkin Least Squares (GLS) FEM and supporting matrix free solver technology that has been applied to Euler and Navier Stokes Analysis (ENSA) of computational fluid dynamics (CFD) problems. The overall capabilities in support of FInite Element Solver for TCAD Applications--will henceforth be referred to as FIESTA.

Problem decomposition is a crucial component in coarse-grain message-passing parallel PDE solvers. To achieve good parallel efficiency, the grid structure describing a particular problem domain must be equitably divided among the processors. Lucas' early 2D work used direct linear solution methods and focused primarily on well-structured problems for which straightforward coordinate bisection and nested dissection algorithms sufficed. This work was extended to 3D grids using iterative linear solution techniques by Wu, et.al [3]. The current parallel version of PISCES, PISCES-MP, has been used to explore several domain decomposition methods for unstructured grids. The method of choice is the Recursive Spectral Bisection (RSB) algorithm of Pothen, et.al. Due to extreme ill-conditioning inherent in the problem discretization, direct linear methods continue to be used and have been extensively benchmarked. Section IV and Appendix B give further details on parallelization of both 2D and 3D versions of the DD formulation of the semiconductor device analysis problem.

III. FIESTA and Benchmarks for the HD Formulation

There are classes of problems where time dependence of the solution and abrupt spatial variations require special consideration and the use of more robust numerical techniques. The field of computational fluid dynamics (CFD) is one such application area. In the TCAD domain, there are both process and device analysis problems where stiffness in time and space is important. In the context of this project, we have focused on analysis of carrier transport in ultra-small devices (i.e. deep submicron MOS and bipolar transistors) where spatially abrupt carrier distributions are of primary importance. For finite volume approaches, substantial effort has been invested in developing upwinding techniques such as the Scharfetter-Gummel approximation [5][6]. While FEM has been applied to semiconductor device analysis [7], the problems with current discretization in 2D have resulted in a hybrid implementation. The objectives of this task were two-fold: 1) to implement the semiconductor device equations in the context of advanced FEM formulation and 2) demonstrate both parallelization and computational benchmarks based on such a prototype code.

A Space-Time and Galerkin Least Squares FEM formulation is developed for the electron and hole HD device equations, and a Galerkin FEM is developed for the Poisson and lattice thermal diffusion equations. One major challenge and accomplishment of this work has been to symmetrize the HD system of equations by employing generalized entropy functions [9]. GLS FEM formulation based on the symmetrized system of equations is shown to satisfy the Clausius-Duhem inequality or the second law of thermodynamics, which is the basic stability requirement for nonlinear system of equations [10]. Appendix A includes the complete discussion so that details are omitted. The governing equations are nondimensionalized to improve conditioning of the system of equations and a staggered approach is employed to treat the coupled HD, Poisson, and lattice equations [11]. As mentioned earlier, since the GLS-FEM has been initially developed in support of CFD applications, shock capturing operators naturally allow for highly nonlinear source terms that occur in the semiconductor problem.

The implementation has followed an SPMD (simple program, multiple data) paradigm to allow for generality as well as parallelization. Again, using leverage provided by work in the CFD community, the ENSA (Euler-Navier Stokes Analyzer) code was chosen as the basis for the solver [8]. The code provides flexibility due to many years of development and applications. Moreover, it runs efficiently on distributed memory, message-passing architectures by exploiting a matrix-free GMRES iterative solution method. Further details of the implementation are provided in the references included in Appendix A [12] [13][14].

In anticipation of creating adaptive FEM solvers, error estimators are required. The major limitation of the existing approaches is in the consistent treatment of the advective terms. In this work, a residual-based asymptotically exact error indicator for elliptic problems that relies on solving local Neumann problems in each element has been extended to the unsymmetric and positive semi-definite advection-diffusion operator. Previous error estimation techniques for the advection-diffusion equation have discarded the advection term in order to stabilize the local error problem. In this formulation, the advection terms are retained and impart stability by including a least squares term. This is a consistent approach since we use the GLS method to solve the global problem.

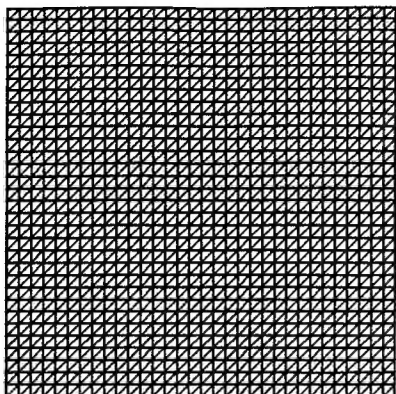
The error is expanded in terms of "bubble" functions which vanish at the nodes. This technique estimates the error as a function of position as opposed to measuring it in a particular norm directly. Thus the analyst has the flexibility of choosing any suitable norm to compute the error indicator. Another advantage of this technique is that the computations are local involving only one or a few neighboring elements at a time, and hence the implementation is almost completely vectorizable and parallelizable.

The element error indicators are used to compute the mesh density function which is input to an advancing front mesh generator to generate the adaptive meshes. To date the testing has been done on 2D problems where exact solutions are available so that the accuracy of the error estimator and the practicality of the refinement strategy can be judged. Figure 2 shows one such example. Another point of particular interest from a numerical perspective is the relationship between the set of PDEs used with the HD formulation and the implication to the boundary conditions that must be imposed. In the course of this work, these factors were carefully considered and documented [15].

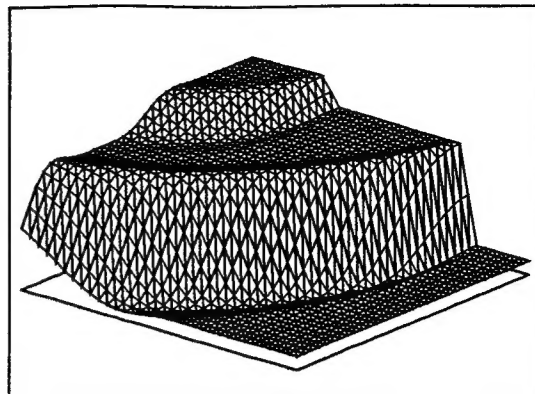
Testing of the FIESTA code involves two aspects: 1) application-oriented evaluation of the results and 2) benchmarking of the parallel performance. It is well-known that $n^+_n n^+$ diodes exhibit thermal non-equilibrium effects near the so-called drain terminal and are widely used as test problems. In this work, both 1D and 2D versions of the $n^+_n n^+$ diode have been analyzed and results compared with others in the literature as well as an implementation of the energy transport formulation in the PISCES code (2ET) [16]. Figure 3 shows a typical 2D velocity profile for $n^+_n n^+$ results. It can be noted that the curves are smooth in both spatial dimensions. Careful comparisons with published results give some indication that for deep submicron structures, analyzed using conventional finite volume discretization of the HD equations, there may be non-physical peaking by as much as 30%. In collaboration with industrial groups such as IBM and ATT, these initial findings are being studied further. However, the primary focus of this project was directed toward the parallel benchmarks discussed below. The simulation of both submicron MOS and bipolar transistors was also demonstrated [12]. Again, the results showed excellent smoothness of solutions with no signs of numerical instability or difficulty in convergence. Further benchmarking and calibration of the results are now being carried out in the computational prototyping project sponsored under a separate ARPA contract. One feature of the GLS approach is the FEM discretization of time as well as space. In the work reported here, the use of the time-dependent aspect of the formulation provides guaranteed stability at the expense of an artificial time-stepping of the solution. On the other hand, in anticipation of the application to TCAD process simulation which is definitely time dependent, the exploration and parallelization of the method provides an excellent platform for further use in the SPRINT-CAD project supported under another ARPA contract.

The parallel benchmarking of FIESTA on both the Intel and IBM parallel machines provides clear evidence of the potential in support of other TCAD applications. Figure 4a shows the speed-up factor for a 2D bipolar problem with 22,000 grid points obtained on the Caltech Delta machine. The example problem demands the solution of a total of 200 million equations over the sequence of time steps in order to obtain a steady-state solution. The parallel efficiency of the HD equations was very close to the ideal results, whereas the Poisson solver portion was much less efficient. However, only the most preliminary parallelization effort was invested on the Poisson equation

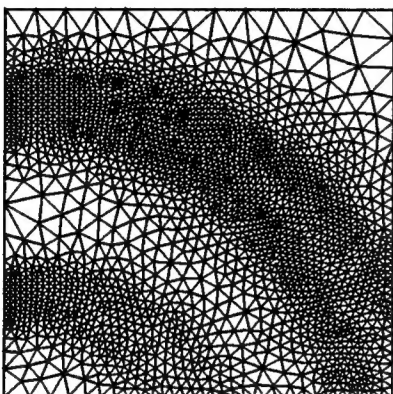
(1089 Nodes ; 2048 Elements)



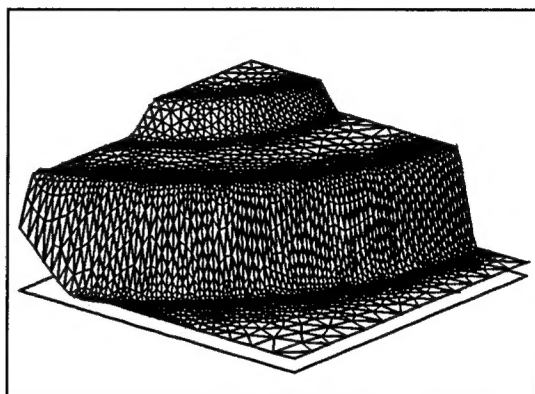
Error = 6.65 %



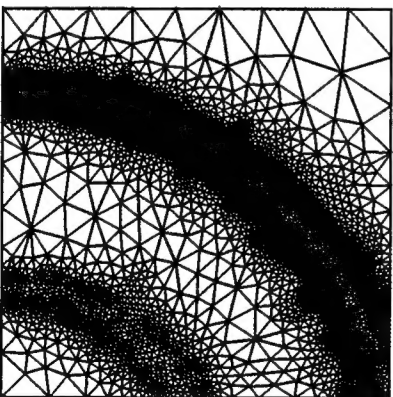
(2061 Nodes ; 4007 Elements)



Error = 2.58 %



(4005 Nodes ; 7896 Elements)



Error = 0.81 %

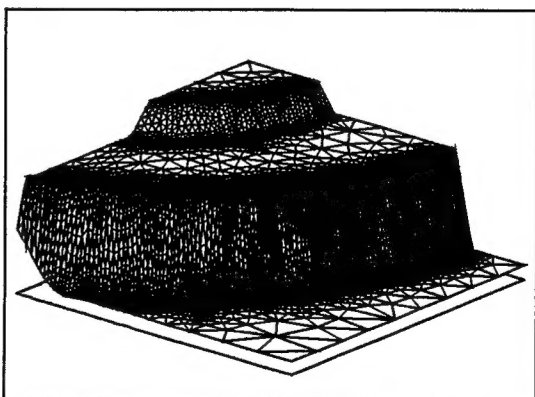


Figure 2: Adaptive meshes and corresponding solutions for a pure advection problem with two interior layers. The error is measured in the L_2 norm and is expressed as a percentage of the norm of the solution.

side and much higher efficiencies are expected in the future. From a practical engineering perspective, the results presented in Figure 4a indicate the ability to achieve an accurate solution in 40 minutes on a 512 node parallel machine. The same problem (and program) when run on an IBM RS 6000/560 fully-loaded workstation took 40 hours. This parallel improvement has both quantitative and qualitative implications--the ability to real-time engineering and innovation depends critically on obtaining timely feedback of information. Figure 4b shows the most recent benchmarks obtained on the 16 node IBM SP/1 machine at Stanford [17]. In this case, the CPU time comparison is made for several interesting device applications on serial and parallel computers. For complicated device structures that require large grid sizes, the serial computers are not only inefficient for practical engineering simulations but are also inadequate. The dramatic reduction in CPU times observed with just 8 processors on IBM SP/1 provides a unique opportunity to perform large-scale device simulations to study device characteristics in ultra small structures. These results again reinforce the engineering importance of the FIESTA code demonstrations and benchmarking. A 16 node machine represents a very practical configuration from an industrial perspective. The parallel performance improvements of more than an order-of-magnitude provide the essential enabling technology that supports real engineering applications.

To summarize the contributions of the FIESTA code and benchmarks, this work has broken new ground in both application of the GLS-FEM technology and in demonstrating near ideal parallel performance improvements based on the ENSA solver implementation. Figure 1 gives a very simplified summary of the FIESTA architecture. The application of FIESTA to submicron MOS and bipolar semiconductor devices has been demonstrated. Further, device-oriented application of FIESTA will occur under the computational prototyping project and the SPRINT-CAD project will make use of the core FEM solver technology.

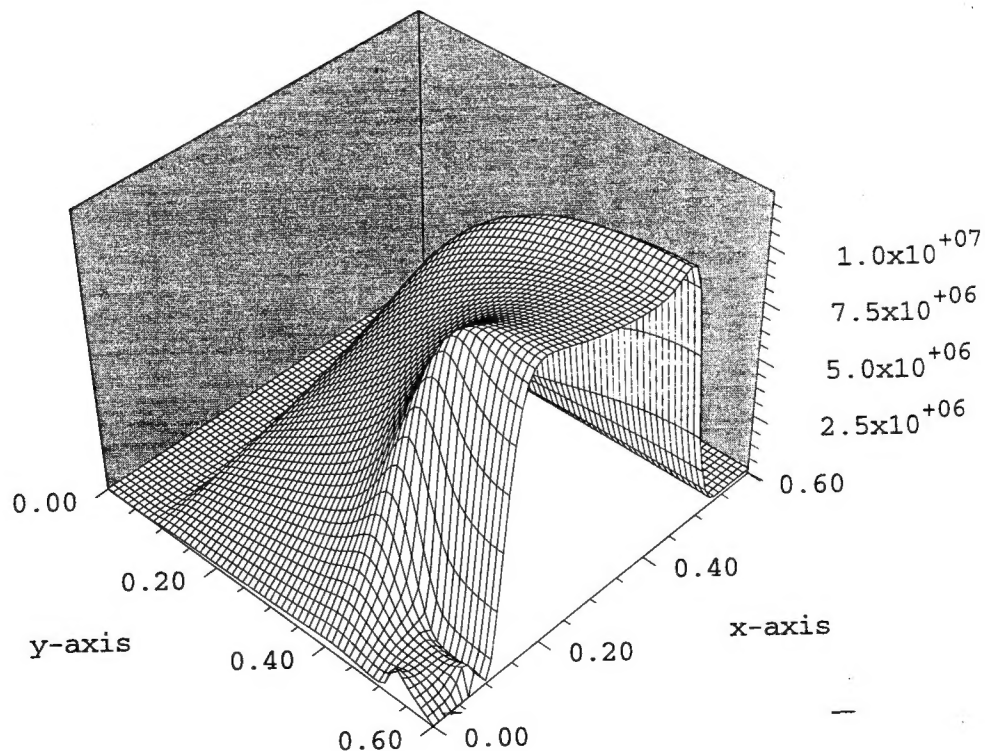


Figure 3: x-component of velocity for a 2D $n^+_n n^+$ diode.

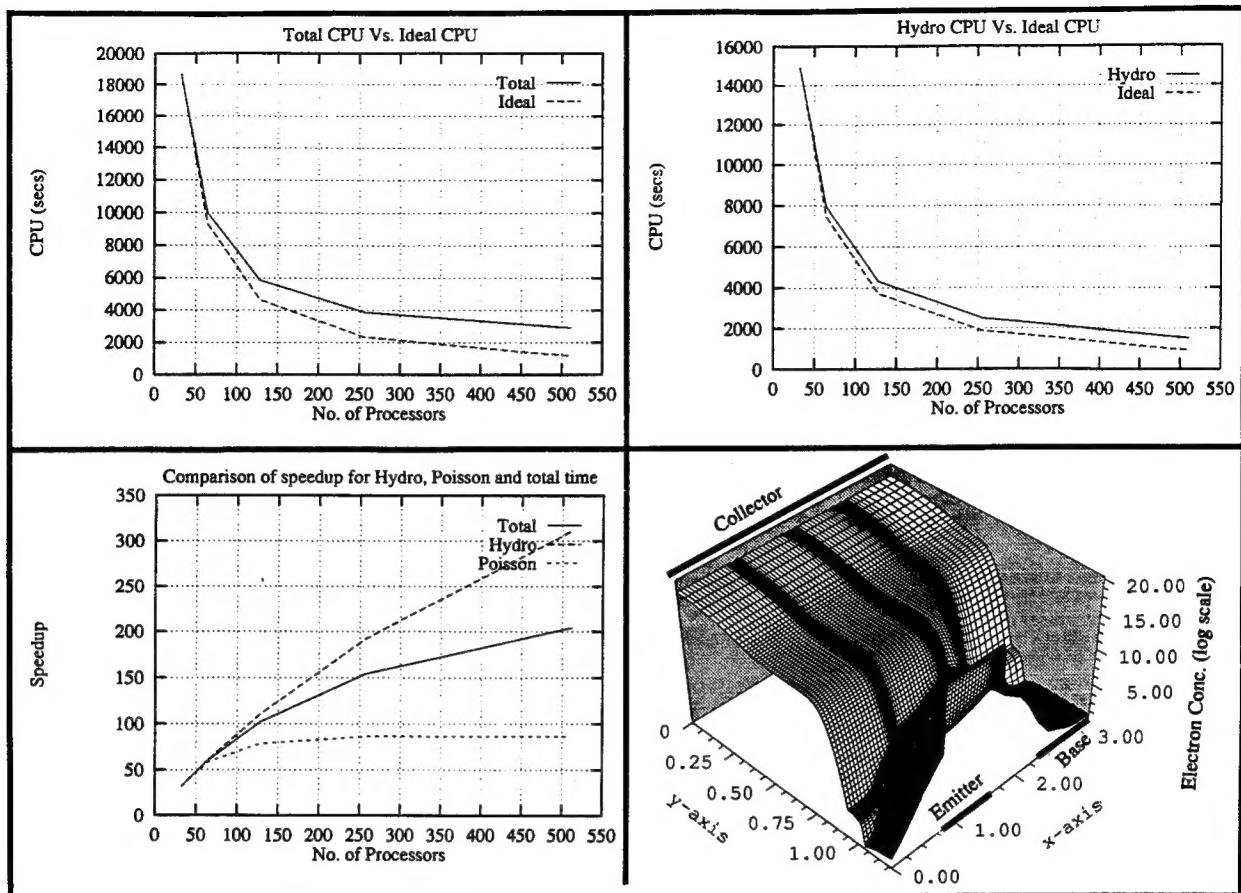


Figure 4a: FIESTA-HD results for bipolar.

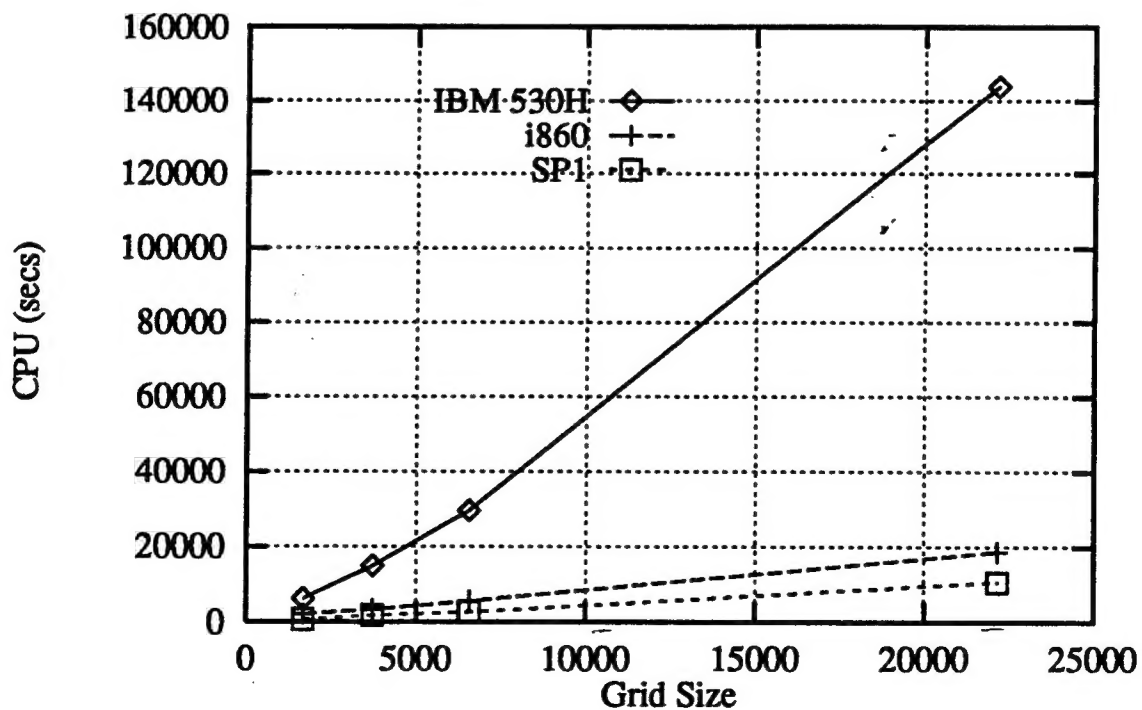


Figure 4b: Comparison of CPU times on IBM 530H, i860 (32 Processors) and SP1 (8 Processors).

IV. Parallel Direct and Iterative Solvers with benchmarks for the DD Formulation

The drift-diffusion (DD) formulation is the most widely used engineering approach for semiconductor device analysis. In contrast to the previous section where advanced FEM technology was used, the section focuses on the parallelization of DD-based application codes with a special emphasis on the parallel linear solvers for sparse matrices. Figure 1 gives a high-level view of both the 2D and 3D prototype codes. As stated in Section II, both incomplete nested dissection and recursive spectral bisection algorithms were integrated into the solvers to perform domain decomposition. By using "plug-in" modules for the domain decomposition, the code can easily use the latest and most promising partitioning algorithms available. Thus, efforts could be concentrated on parallelizing the existing semiconductor device analysis applications and their linear solvers.

The PISCES code (release 2B, version 9009) was used as the test vehicle for the 2D parallelization. PISCES is an widely-used, industry-standard code with over 15 years of development history. The code encompasses many areas of interest in both physics and computational mathematics. Almost all aspects of this complex code have been parallelized including physical model evaluations, matrix formation and assembly, non-linear solution, and linear solution. A complete discussion of the creation of the parallel application, PISCES-MP, has been presented in [13][18][19] and is included in Appendix B for completeness. To facilitate the parallelization of PISCES, the existing public domain linear solver was replaced with a parallel direct linear solver based on the work of Lucas [1]. The resulting parallel application has been ported to three parallel platforms: Intel, Thinking Machines, and IBM. The distributed-memory, message-passing parallelization methodology used made porting among the architectures straightforward. In fact, a major achievement of the PISCES-MP project has been to understand in detail, by means of consistent benchmarking, the aptitudes of the various parallel machines. Moreover, since PISCES-MP is a fully-functional application with broad acceptance in the IC technology community, these results carry user-side credibility on promoting and accelerating use of parallel machines. The benchmarks presented below are primarily a sample with further details given in Appendix B.

PISCES-MP benchmarks have been created and run for both MOS and bipolar problems. Over the course of this work, the creation of adequate 2D grid has been a non-trivial problem. Moreover, as the benchmark sizes continue to increase, the limitations of the PISCES code itself have been stretched, broken, and repaired. Figure 5a shows the cross-section of a CMOS inverter structure (both n- and p-channel devices). Figure 5b shows the resulting 16-way decomposition generated using recursive spectral bisection. This problem is indicative of many VLSI applications where complex cross-sections of multiple (and often interactive) devices are key aspects of the circuit design process. Figure 5c shows the comparative benchmark solution times for a series of increasingly fine grid resolutions. Solving for the potential and both carrier concentrations yields linear systems three times the grid size. The grids were run on a 32-node Intel iPSC/860, a 32-node TMC CM-5E (without vector units), a 16-node IBM SP1, and a 16-node IBM SP2. For comparison, the runtime on an IBM RS/6000 Model 560F, the fastest serial computer available, is also shown. Due to insufficient memory, the largest problem could not be run on the iPSC/860 or the workstation. The superior performance of the parallel machines becomes evident at even mod-

erate grid sizes. Of special interest for both IBM machines is the excellent performance achieved with relatively modest numbers of processors - an important factor to be considered from the perspective of engineering applications where cost/performance is at a premium.

The second benchmark problem is targeted at exploring the limits of problem size and machine capabilities for solving truly huge 2D problems. The device is a simplified version of a multilayer structure that is a light-emitting diode (LED) used for optical data communications. This structure is particularly challenging since the resolution required for atomic layer features can easily use hundreds-of thousands of grid points. Figure 6 shows the solution times for a structure with 112,000 grid points (336,000 equations) on the IBM SP-2. Due to the immense resource demands, only larger machine partitions with sufficient aggregate memory are capable of solving this problem. Clearly, when problem resource demands are sufficient, large numbers of high-performance processing nodes can be efficiently utilized. With 128 processors, the computation sustains almost 6.5 GFLOPS with over 70% efficiency.

The solution of 3D device analysis problems is generally believed to be beyond the practical capabilities of direct solvers. The STRIDE code has been developed as a test vehicle to explore parallel iterative solvers in the context of the DD formulation [2][3]. While all the code development was done on Intel computers, during the last few months of this research contract, core routines were ported to and benchmarked on the SP/1 as well. A broad range of parallel matrix solution methods and preconditioners that work well for the semiconductor problem have been demonstrated. Concurrent ILU(1) and ILUV preconditioners have been developed and demonstrated to be effective where concurrent ILU(0) is insufficient. Results indicate that ILUV is more promising for difficult problems and can achieve convergence with much less fill-in than for ILU(0). These results indicate that for ill-conditioned matrices, the condition number of the preconditioner can become a dominant factor in determining effectiveness. While the primary results in this study were demonstrated using the STRIDE code with tensor product grid, a prototype version of an object-oriented sparse matrix solver was also demonstrated. The single processor version achieved identical performance to the FORTRAN version used with STRIDE. In addition, benchmarks on an industrial matrix problem provided by IBM will be discussed in the final section of the report.

The STRIDE benchmarks on parallel iterative solvers for the DD formulation of a bipolar transistor with 4.9 million grid was demonstrated on the Caltech Delta machine. For the 512 node machine, a maximum sustained performance of 1.7 GigaFlops was achieved, leading to recognition in the Grand Challenges competition in 1992. Of equal and even greater importance is the fact that convergence of these huge 3D bipolar problems was achieved in an average of 20 minutes per bias point. This is indeed an encouraging result that suggests scaling to still larger problems will be very practical based on more powerful processors than are currently available. One rather surprising result was the fact that the growth of computational time with grid size was a very modest 1/3-power law. This is a result of the careful management of communication requirements and the fact that with larger problems, the processors have more work to keep them busy and hence are communicating less.

An important postscript to the benchmarking of both PISCES-MP and STRIDE codes is the fact that memory size and its full utilization are key aspects of achieving high performance parallel

computing. Namely, having high-performance computational nodes with commensurate per-node memory is of prime importance to effectively realize parallel computing's potential. In terms of the benchmarks themselves for a fixed problem size and moving to more parallel processors, there comes a point of diminished returns where less work is given to each processor and the communications requirements (and their frequency) start to overcome benefits of having more processors. Hence, the results presented in this section, while indeed promising, cannot be considered definitive in showing scaling limits for parallel applications in semiconductor device analysis. Quite to the contrary, much of the limitations revealed here press the need for more robust grid generation and the more complete restructuring of the applications themselves. Finally, there are indeed opportunities to improve the domain decomposition process, both for static and dynamic grid conditions.

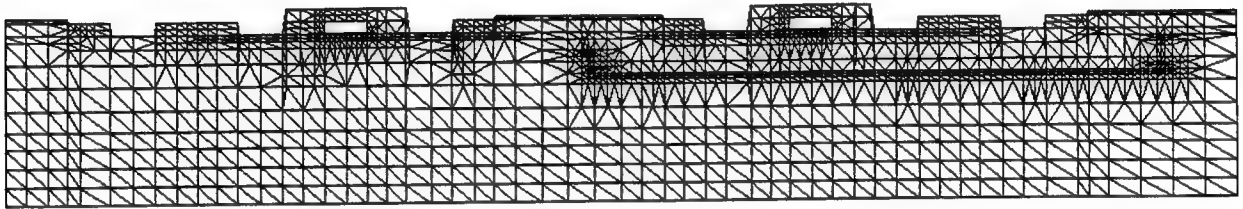


Figure 5a: Grid structure of a CMOS inverter.

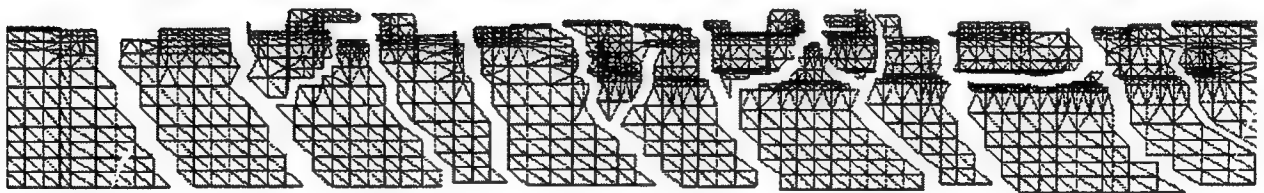


Figure 5b: 16-processor decomposition using Recursive Spectral Bisection.

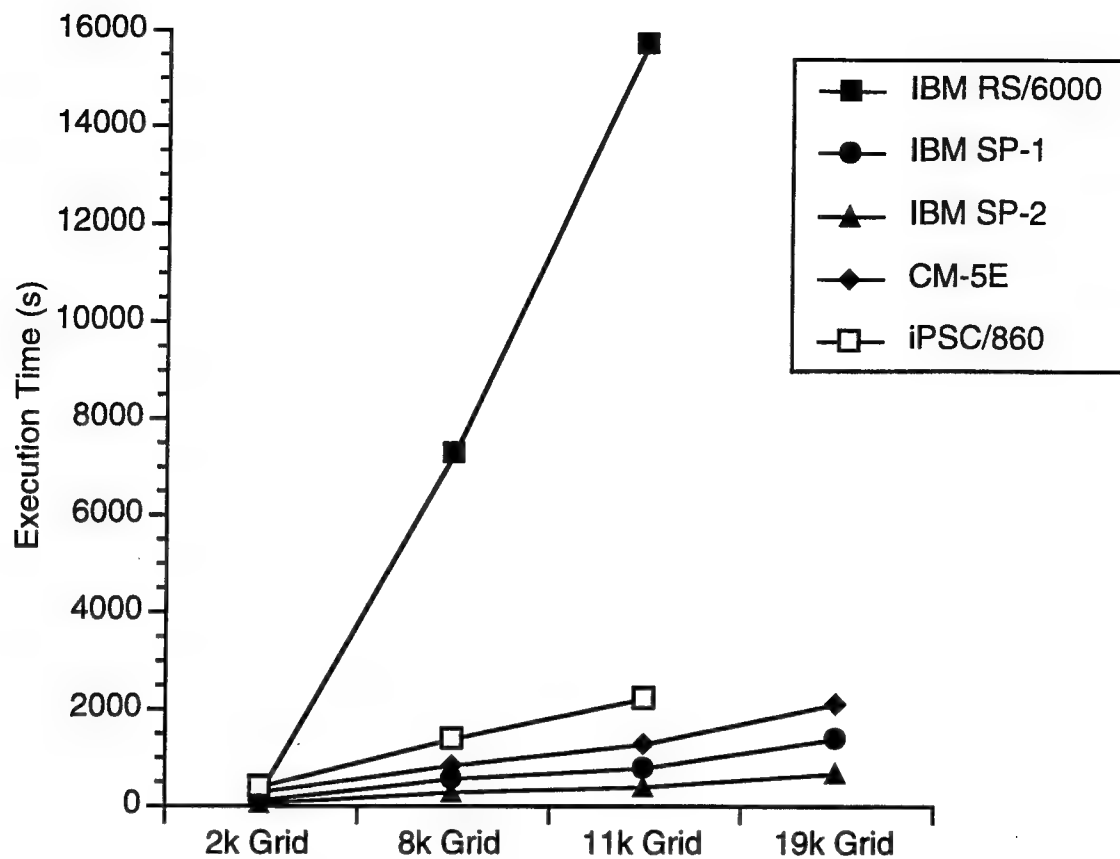


Figure 5c: Comparison of CMOS inverter simulation on several architectures.

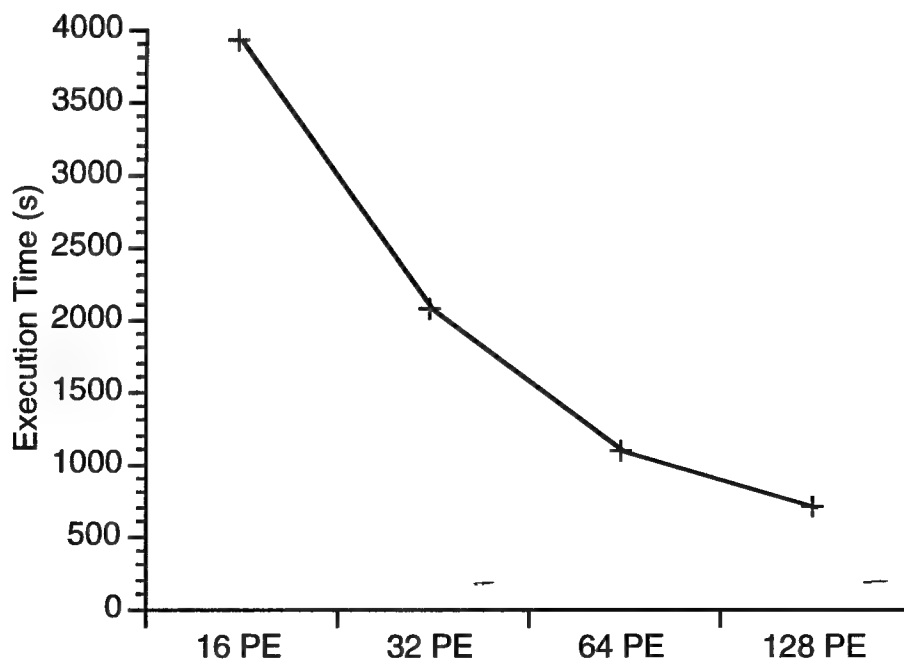


Figure 6: Photo-emitter simulation with 112,000 grid on IBM SP-2.

V. Integrated Geometry and Grid Services in Support of 3D TCAD

In order to support 3D TCAD applications using parallel computational capabilities, there are a number of infrastructure issues that need to be addressed in both specifying 3D structures and in gridding them for simulation. The full extent of the challenge involved in both these tasks was not apparent at the outset of this project. While the STRIDE examples with millions of grid in 3D had been achieved early on in the project, the task of generalizing these capabilities to nonplanar surfaces proved to be major effort of its own proportions. Moreover, in the early stages of the project, there was considerable optimism that a standardized semiconductor wafer representation (SWR) for 3D would progress at a rate that would interleave with requirements for this project. In fact, over the course of this project, the SWR efforts became stalled at both a committee level and in terms of actual prototyping. The results presented here have achieved significant advances in developing both a unified representation and supporting tools for both 2D and 3D gridding services in support of a TCAD framework, despite the missing infrastructure in terms of a full SWR. Nonetheless, it is important to note that such development efforts are beyond the scope and manpower of this project.

The specification of solid geometry services to support a fully 3D wafer representation were initially proposed (SWR, Version 1.5) and subsequently revised in the course of this project [20]. Based on this information model, a commercial solid modeler from Spatial Technology called ACIS was used to create a fully functional 3D geometry server that was used for two applications--as a 3D structure generator as well as a support utility for 3D gridding of nonplanar structures. The gridding aspects are discussed shortly. The purely geometric use of the server resulted in a new application code for creating 3D Virtual Integrated Processes (VIP3D) starting with IC mesh layout information and a minimal set of process parameters such as layer thicknesses and shape factors (i.e. gate spacers, locally oxidized isolation shapes, etc.). The results of such a purely geometric representation of a 3D IC structure are illustrated in Figure 7 which shows an array of 16 cells each containing four transistors static memory (SRAM) gates. Further details of some of the layers are also shown. It can be seen that complex features of the design are readily mimicked in the computational prototype. The SRAM design considered in Figure 7 came from a commercial development project at Cypress Semiconductor. In fact, a simplified SRAM design was used as a vehicle to test a suite of TCAD tools to extract IC behavior at the circuit level based on layout information and simple technology parameterization [21]. Among the most mature and promising of the applications that can directly use results of VIP3D are: 1) interconnect analysis of complex 3D structures based on the multiple analysis technique used in FASTCAP [22] and 2) mixed-mode analysis of arbitrary technology cross-sections where 2D device analysis such as PISCES can be used in concert with the SPICE circuit analysis program [16] practical test conditions can be imposed on multi-device cross-sections such as that used for illustration in the last section (see Figure 5a). These demonstrations are very recent and the prototype of VIP3D and supporting applications are immature. Nonetheless, the industry response to these demonstrations has been highly enthusiastic. It is recommended that further attention and effort be applied in creating more complete infrastructure for 3D TCAD based on this prototype.

The gridding support for 3D TCAD is a major challenge. As indicated earlier, the specification of a viable information model for 3D has received only modest support through a SEMATECH Lithographer's Workbench (LWB) project. In fact, in a recent workshop sponsored by SEMAT-

ECH, a proposal for implementing an SWR Version 2.0 was turned down by the sponsoring companies. A major problem with both the Version 1.5 document and the Version 2.0 proposal is the over specification of nonessential features which make implementation difficult. In this project, two distinctly different implementations of 2D and 3D gridding capabilities have been created using a subset of the SWR, Version 1.5 specification. In both cases, a tree-based data structure (quad-tree for 2D and oct-tree for 3D) has been used. The key feature of the tree-based approach of interest for this project is the opportunity to use the tree as a logical structure for refinement as well as coarsen of the grid. In the context of implementing the gridders based on a client-server architecture, both codes have used geometry operations to handle various issues related to interfaces and surfaces. The following discussion summarizes each of the gridders used and developed in this work.

The FOREST girder was developed in support of 2D process and device simulation research funded by the Semiconductor Research Corporation (SRC). As stated above, the quad-tree approach is used and a custom geometry server capability is directly integrated into the code. Both static and dynamic (moving) grids are handled by FOREST. While it is beyond the scope of this project to provide detailed documentation of the code and algorithms, Appendix C includes several papers [23][24] that summarize the capabilities. Of special interest in the context of this project is the fact that problems suitable for parallel analysis with PISCES-MP can be generated either using FOREST or the 2D/3D girder described next. Because FOREST was very specifically targeted to support 2D analysis, it is more aggressive in its algorithms and its use of grid to resolve fine features in IC cross-sections.

The CAMINO girder has evolved out of the thesis work of Yang [25] with the specific intent to handle complex 3D structures. The gridding and refinement algorithms are based on quad- and oct-tree methodology. The development of a level-control function [23] was one key innovation needed to avoid the excessive refinement of grid in areas not of practical interest. The second major contribution has been the implementation of a delta-zone at surface regions in order to support refinement at non-planar surfaces and still maintain the tree data structure without excessive refinement [26]. In contrast to the built-in geometry services used in FOREST, the CAMINO program uses the same ACIS-based server described earlier as part of VIP 3D. This choice in implementation was targeted in testing the approach and algorithms and not for efficiency. In the course of industrial collaboration with IBM as discussed in the next section, the use of ACIS is now being reconsidered. In fact, jointly with IBM a different solid geometry modeler was interfaced to further test and benchmark CAMINO. As part of follow-on work in the SPRINT-CAD project, an efficient special purpose geometry server will be implemented in CAMINO.

In summary, several key utilities and framework services were developed as part of this project. While it was initially anticipated that standardized SWR services could be used in this project, this turned out not to be possible owing to factors outside the scope and resources of this project. However, the use of three prototype codes in support of this project have clearly demonstrated the viability of both the client-server model and the constituent geometry and grid information models. The VIP 3D code based on purely geometric specification of 3D device structures has been shown to be a very practical and powerful technique. In addition, the FOREST and CAMINO tools have created a solid foundation for semi-structured 2D and 3D gridding respectively using a tree-based algorithmic approach. Both these tools are capable of supporting hierarchical grid refinement and will be developed as part of the SPRINT-CAD project.

VI. Technology Transition including Industrial Applications

The application of TCAD to the design and manufacturing of ICs has demonstrated major benefits and cost savings that have been quantified by virtually all manufacturers in the industry. The use of 1D and 2D simulators has been the workhorse of the industry. Yet the need for more powerful 3D tools has emerged as a pressing one with the drive for higher frequency performance and scaling of technology to deep submicron structures. In the course of this project, both practical examples and industrial interactions have helped to drive the development and to quantify the output. Several of these points have been mentioned in the previous sections and are now discussed in greater detail.

In the period from Fall 1992 through Spring 1993, the ALADDIN-CAD project (AnaLysis of ADvanced Devices based on Industry-Networked TCAD) had been a consortia driven effort to apply and test early prototypes of parallel software for TCAD. The final report on that project [27] provided documentation of both the technical achievements and the industrial interactions that lead to follow-on collaborations are key parts of the work reported here. Specifically, the following companies and their application domains each contributed to the testing and benchmarking of results from this follow-on project:

- Cypress Semiconductor--SRAM development based on 3D geometry modeling and prototype components of VIP3D.
- Hewlett-Packard--High-speed device design and optimization with emphasis on optoelectronics and high-speed components for data communications--test examples for PISCES-MP came from these collaborations.
- IBM--Testing and applications of virtually all aspects of this project. The donation of a 16 node SP/1 to Stanford in 1993 for research collaboration allowed key benchmarks to be quickly developed and reported.
- Intel--Application of the 3D STRIDE code for parasitic analysis was an early example of code transfer to industry and its ongoing industrial use.
- National Semiconductor--High voltage semiconductor analysis and design has provided another example and benchmark for the PISCES-MP code.

The details of Cypress Semiconductor's application of 3D geometry modeling are proprietary. On the other hand, Figure 7 clearly shows that practical examples can now be created and used effectively in industrial practice. In fact, Dr. Tony Alvarez, VP of Research and Development, has provided us with key feedbacks about both the benefits and an estimate of the quantitative benefits of this work (see letter in Appendix D). As mentioned in the previous section, while the VIP3D software is still very much in a prototype stage, it is clear that further development and application are of the highest potential. In fact, a unique strength of the approach used in this work is its applicability to a wide range of technologies (i.e. OEIC, MEMS, etc.) in addition to the primary demonstration in the arena of VLSI silicon.

The interactions with HP, Intel, and National Semiconductor each represent a sampling of diverse applications and the unique leverage possible with large scale device simulations. In the cases of both HP and National, the device structures each put major demands on analysis capabilities of PISCES-MP. The fact that the scaling of problems sizes beyond 100,000 grid level and the robust

and efficient performance of the direct solver used are most encouraging. In fact, the direct solver technology has been sought after by vendors not only from the TCAD domain but also those involved in parallel applications to mechanical engineering applications. Specifically, the Centric company intends to use algorithms developed from this work as an integral part of a follow-on contract with IBM, under ARPA support. This clearly demonstrates and validates the claim that spin-on technology applications other than for TCAD have resulted from this work.

The Intel applications of STRIDE on their parallel machine and in support of industrial 3D IC design considerations has been a major benefit of the ALADDIN-CAD project that has carried over into this work. The letter of Dr. Francisco Leon, Program Manager of Process and Device Modeling included in Appendix D, gives clear evidence of the ongoing impact of the work. Of special interest and importance is the collaboration related to 3D geometry-based modeling. In spite of the difficulties arising out of the delayed development of the SWR, Intel has been a key driving force in looking for viable technical approaches that meet the long-term needs for 3D TCAD. It is expected that follow-on efforts related to the SWR in the context of the SPRINT-CAD project will bear fruit over the next year.

The interactions with IBM are of special importance. In some sense, the impact may show the greatest potential for long-term growth. The ALADDIN-CAD support letter from IBM (see Appendix D) indicates that not only has the collaborative research been an excellent model for interaction but, additionally, that IBM has chosen to further leverage the research efforts through substantial equipment donations. In the case of this project, the donation by IBM of a 16 node SP/1 computer has made it possible to accelerate not only the development and benchmarking of parallel TCAD software, it has set the stage for other industrial and research applications of the technology. As noted in both sections III and IV of the report, excellent benchmark results for both FIESTA and PISCES-MP have been achieved on the SP/1 at Stanford as well as the SP/2 at the NASA/NAS facility. In fact, over both this project and the Aladdin-CAD efforts, the availability of both a local machine at Stanford and more powerful scaled-up version at NASA/NAS has been a highly effective means to develop benchmark and further test the limits of research codes.

The breadth of the IBM interactions, stimulated through this project as well as ALADDIN-CAD are beyond the scope of the present report. However, a few more highlights will serve to illustrate their importance and potential impact:

- Collaborative efforts to apply both more advanced HD modeling and algorithms for parallelization of the FIELDAY program.
- Benchmarking of large sparse problems provided by IBM to test Stanford's iterative solvers based on parallel preconditioning.
- Collaboration in porting the CAMINO 3D gridding tool into the IBM environment, based on their internal solid geometry support utilities, and to test its application for industrial problems.
- Porting of IBM's FIELDAY program to Stanford for application in the ARPA-related project--both SPRINT-CAD and the new Computational Prototyping project.
- Assistance in adapting FIELDAY to support the mixed-mode capabilities for coupled device-circuit analysis in conjunction with a Stanford-modified version of UC Berkeley SPICE.

- Collaboration in providing Stanford a license to use the DAMOCLES Monte Carlo device analysis code in support of both the ARPA Computational Prototyping project and the NSF National Center for Computational Electronics (NCCE).
- Collaboration in producing visualization results for the 4.9 million bipolar transistor examples computed on the Intel Delta machine.

These collaborations with IBM were facilitated in large part through the participation of Dr. Ronald Knepper of IBM East Fishkill, an industrial visitor at Stanford's Center for Integrated Systems. The interactions have involved more than a half-dozen collaborators at five locations within IBM. As noted in the ALADDIN-CAD support letter from IBM, the fostering of such a relationship from the initial consortia efforts has been of major benefit to both the Stanford research efforts and to IBM as an industrial partner.

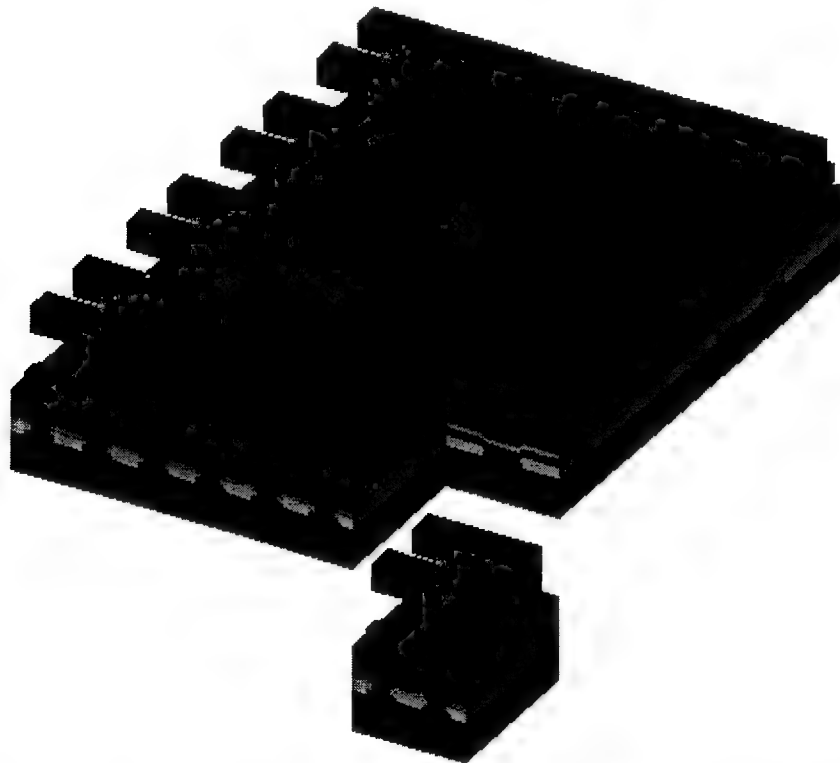


Figure 7: 3D modeling of 4x4 SRAM cell array. Each cell consists of 4 transistors and polysilicon lines as load resistors. Shown in the figure are two global metal lines, the lower one being the bit lines. A stand-alone cell is also shown.

VII. Conclusion

The previous sections have summarized the results of a three-year research effort to demonstrate the parallelization of FEM Software for semiconductor device simulation. The results clearly demonstrate both the viability and computational efficiency of such parallel codes applied to the semiconductor domain. In addition, there is clear potential that the availability of such codes can have a long-term impact on the IC industry to achieve greater efficiency and effectiveness in the design of high performance hardware.

The specific achievements of this project can be quickly summarized as follows:

- development and parallelization of a GLS-FEM formulation of the HD transport equations using matrix-free GMRES;
- parallel benchmarks of the GLS-FEM code on Intel and IBM machines for 2D MOS and bipolar examples;
- parallelization of a mature 2D finite volume code PISCES based on a direct solver and its porting across Intel, TMC, and IBM machines;
- benchmarks of the PISCES-MP for 2D examples scaled with available memory to 128 processors and hundreds-of-thousands of grid, achieving 6.5 GFlops on an SP/2;
- implementation of parallel preconditioned iterative solvers using ILU(0), ILU(1), and ILUV and complete application testing using the STRIDE code based on the DD formulation;
- parallel benchmarks using STRIDE for 3D bipolar examples with 4.9 million grid computed on the 520 processor Delta machine, achieving convergent solution in 20 minutes per bias point;
- development of quad- and oct-tree gridding utilities in support of parallel TCAD applications;
- development of a 3D geometry-based structure generator based on a Virtual Integrated Process (VIP 3D) representation;
- testing of several large-scale industrial examples for PISCES-MP including: CMOS gates, high-voltage and optoelectronic device structures;
- technology transfer of algorithms, code fragments and even full applications to industrial partners.

References:

- [1] R. F. Lucas, "Solving Planar Systems of Equations on Distributed-memory Multiprocessors," Ph.D. Dissertation, Stanford University, 1987.
- [2] K-C Wu, R.F. Lucas, Z. Wang, and R.W. Dutton, "New Approaches in a 3-D One -carrier Device Solver," *IEEE Trans. CAD*, vol. 8, no. 5, pp. 528-537, 1989.
- [3] K-C Wu, G. Chin, and R.W. Dutton, "A STRIDE Towards Practical 3-D Device Simulation-- Numerical and Visualization Considerations," *IEEE Trans. on CAD*, vol. 10, no. 9, pp. 1132-1140, 1991.
- [4] H.D. Simon, "Partitioning of Unstructured Problems for Parallel Processing," *Comp.Sys. in Eng.*, vol. 2, pp. 135-148, 1991.
- [5] M. Pinto, C. Rafferty, H. Yeager, and R.W. Dutton, "PISCES-IIB Supplementary Report," Stanford University, 1985.
- [6] R.W. Dutton and Z. Yu, *TCAD--Computer Simulation of IC Processes and Devices*, Kluwer Academic Publishing, 1993, pp. 373.
- [7] E.M. Buturla, J. Johnson, S. Furkay, and P. Cottrell, "A New Three-dimensional Device Simulation Formulation," NASECODE VI: Proceedings of the Sixth International Conference on the Numerical Analysis of Semiconductor Devices and Integrated Circuits, J.J.H. Miller, Ed., Boole Press Ltd., Dublin, 1989.
- [8] F. Shakib, T.J.R. Hughes, and Z. Johan, "A New Finite Element Formulation for Computational Fluid Dynamics: X. The Compressible Euler and Navier-Stokes Equations," *Comp. Methods in Applied Mechanics and Eng.*, vol. 89, pp. 141-219, 1989.
- [9] N.R. Aluru, A. Raefsky, P.M. Pinsky, K.H. Law, R.J.G. Goossens, and R.W. Dutton, "A Finite Element Formulation for the Hydrodynamic Semiconductor Device Equations," *Computer Methods in Applied Mechanics and Engineering*, vol. 107, pp. 269-298, 1993.
- [10] N.R. Aluru, K.H. Law, A. Raefsky, P.M. Pinsky, R.J.G. Goossens, and R.W. Dutton, "Numerical Solution of Two-carrier Hydrodynamic Semiconductor Device Equations Employing a Stabilized Finite Element Method," to appear in *Computer Methods in Applied Mechanics and Engineering*.
- [11] N.R. Aluru, K.H. Law, P.M. Pinsky, R.J.G. Goossens, and R.W. Dutton, "Space-Time Galerkin/Least Squares Finite Element Formulation for the Hydrodynamic Device Equations," *IEICE Trans. Electron.*, vol. E77-C, no. 2, pp. 227-235, 1994.
- [12] N.R. Aluru, K.H. Law, and R.W. Dutton, "Simulation of the Hydrodynamic Model on Distributed Memory Parallel Computers," in preparation.
- [13] B.P. Herndon, N.R. Aluru, A. Raefsky, R.J.G. Goossens, K.H. Law, and R.W. Dutton, "A Methodology for Parallelizing PDE Solvers. Application to Semiconductor Device Simulation," VIIth SIAM Conference on Parallel Processing for Scientific Computing, Feb. 1995, San Francisco, CA.
- [14] K.H. Law, "Large Scale Engineering Computations on Distributed Memory Parallel Computers and Distributed Workstations," NSF Workshop on Scientific Supercomputing, Visualization, and Animation in Geotechnical Earthquake Engineering and Eng. Seismology, Nov. 1994.
- [15] N.R. Aluru, K.H. Law, P.M. Pinsky, and R.W. Dutton, "An Analysis of the Hydrodynamic Semiconductor Device Model--Boundary Conditions and Simulations," submitted for publication.
- [16] Z. Yu, D. Chen, L. So, and R.W. Dutton, "PISCES-2ET--Two Dimensional Device Simula-

- tion for Silicon and Heterostructures," Stanford University, 1994.
- [17] N.R. Aluru, K.H. Law, A. Raefsky, and R.W. Dutton, "FIESTA-HD: A Parallel Finite Element Program for Hydrodynamic Device Simulation," submitted to parallel CFD'95, California Inst. of Technology, Pasadena, CA, June 26-28, 1995.
 - [18] B.P. Herndon, A. Raefsky, and R.J.G. Goossens, "PISCES MP--Adaptation of a Dusty Deck for Multiprocessing," Proceedings of NASECODE-VII, May, 1992.
 - [19] B. Herndon, A. Raefsky, R. Goossens, and R.W. Dutton, "A Methodology for Parallelizing PDE Solvers: Applications to PISCES," submitted to the *Journal of Computer and Software Engineering, Special Issue on Parallel Computing*.
 - [20] TCAD Framework Group, Semiconductor Wafer Representation Working Group, *Semiconductor Wafer Representation Procedural Interface (PI)*, version 1.5, Technical Report, CAD Framework Initiative Inc., 1993.
 - [21] K. Wang, F. Rotella, T. Chen, D. Yang, A. Lee, Z. Yu, R.W. Knepper, J. Watt, and R.W. Dutton, "Layout-based Extraction of IC Electrical Behavior Models," IEDM 1994 Proceedings, December 1994, San Francisco, CA, pp. 209-212.
 - [22] K. Nabors and J. White, "A Fast Multiple Algorithm for Capacitance Extraction of Complex 3-D Geometries," Proc. Custom Int. Circuits Conference, San Diego, CA pp. 21.7.1-21.7.4.
 - [23] Z. Sahul, E. McKenna, R.W. Dutton, "Grid Evolution for Oxidation Simulation Using a Quadtree Based Grid Generator," NUPAD V Conference, June 5-6, 1994, Honolulu, HI, pp. 155-158.
 - [24] Z. Sahul, E. McKenna, R.W. Dutton, "Grid Techniques for Multi-Layer Device and Process Simulation," TECHCON'93, Sept. 28-30, 1993, Atlanta, GA.
 - [25] D. Yang, "Mesh Generation and Information Model for Device Simulation," Ph.D. Dissertation, Stanford University, 1994.
 - [26] D. Yang, K. Law, and R.W. Dutton, "An Automated Mesh Refinement Scheme Based on Level-Control Function," Proceedings of NUPAD IV, May 31-June 1, 1992.
 - [27] R.W. Dutton, "Analysis of Advanced Devices Using Industry-Networked Technology CAD (ALADDIN-CAD)," Final Report, California Competitive Technology Program (CompTech Grant# C90-072).

A finite element formulation for the hydrodynamic semiconductor device equations

N.R. Aluru^a, A. Raefsky^b, P.M. Pinsky^a, K.H. Law^a, R.J.G. Goossens^b and R.W. Dutton^b

^a*Department of Civil Engineering, Stanford University, Stanford, CA 94305, USA*

^b*Department of Electrical Engineering, Stanford University, Stanford, CA 94305, USA*

Received 21 September 1992

A new formulation employing the Galerkin/least-squares finite element method is presented for the simulation of the hydrodynamic model of semiconductor devices. Numerical simulations are performed on the coupled Poisson and hydrodynamic equations for one carrier devices. The hydrodynamic equations for a single carrier, i.e. for the electrons or holes, resemble the compressible Navier–Stokes equations with the addition of highly nonlinear source terms and without the viscous terms. The governing equations are nondimensionalized to improve the conditioning on the resulting system of equations and the efficiency of the numerical algorithms. Furthermore, to establish the stability of the discrete solution, the system of hydrodynamic equations is symmetrized by considering generalized entropy functions. A staggered solution strategy is employed to treat the coupled hydrodynamic and Poisson equations. Numerical results are presented for one-dimensional and two-dimensional one-carrier $n^+ - n - n^+$ devices. The presence of velocity overshoot has been observed and it is recognized that the heat flux term plays an important role in the simulation of semiconductor devices employing the hydrodynamic model.

1. Introduction

The simulation of the electrical characteristics of semiconductor devices has been an active area of research for over a decade. Such research has led to the development of a series of increasingly powerful and full-featured simulators [1]. Recent advancements in three distinct areas have created the opportunity for another round of breakthrough developments.

Firstly, continued device miniaturization has pushed geometry sizes down to below $0.5\ \mu\text{m}$, leading to ever higher electrical fields. Therefore, it is no longer reasonable to assume a simple linear relationship between carrier velocity and local electric field. Instead, more complicated models are needed to explicitly deal with this carrier-heating phenomenon. As a result, there has been a shift away from the commonly used drift-diffusion model. The two main contenders are the energy-transport [2] and the hydrodynamic model [3–7]. The latter set of equations obtains its name from the strong similarity to the compressible Euler and Navier–Stokes equations governing fluid flow. Current simulators dealing with the hydrodynamic

Correspondence to: Professor Kincho Law, Stanford University, Department of Civil Engineering, Terman Engineering Center, Room 242, Stanford, CA 94305, USA.

(HD) model have been rather restricted: many only deal with 1-D problems, all rely on ad-hoc heuristic numerical ‘tricks’ to help the solution process, none have systematically dealt with verification of correctness.

Secondly, in the area of computational fluid dynamics, developments over the past 5 years in the Galerkin/least-squares finite element formulation of compressible Euler and Navier–Stokes equations have led to very general, robust, and accurate codes [8–10, 15, 19, 21, 22]. To our knowledge, there is no literature employing these methods to the hydrodynamic model for the semiconductor device equations.

Thirdly, implementations of the Galerkin/least-squares finite element method map nicely onto modern massively parallel architectures as has been demonstrated through the solution of million-element problems on highly unstructured grids [11]. This has made it possible to attack interesting engineering problems with a realistic degree of complexity and produce solutions within a reasonable time.

In this paper, we propose a space-time Galerkin/least-squares finite element method based on the hydrodynamic model for semiconductor devices. Coupled hydrodynamic and Poisson equations are solved using a staggered scheme. The non-symmetric, nonlinear hydrodynamic equations are symmetrized with generalized entropy functions. This formulation based on entropy variables automatically satisfies the Clausius–Duhem inequality, or the second law of thermodynamics, which is a basic nonlinear stability requirement. To improve the conditioning of the resulting system of equations, the governing equations are nondimensionalized.

The paper is organized as follows. In Section 2, we review the partial differential equations for the hydrodynamic model and the Poisson equation, establish similarity between the HD equations and the compressible Euler equations, and discuss nondimensionalization procedures. In Section 3, we give the conservation form and present the symmetrization procedure. Section 4 discusses the finite element formulation of the electron hydrodynamic equations and the Poisson equation. Section 5 discusses the staggered approach that we use to solve the coupled equations. In Section 6, we present the numerical results for one-carrier devices to demonstrate the robustness and applicability of finite element methods for device simulations. In Section 7, we summarize the contributions of this study and future research.

2. Partial differential equations for semiconductor devices

Semiconductor devices can be simulated by solving a set of conservation equations for the electrons and holes coupling with the Poisson equation for the electrostatic potential. The partial differential equations for the conservation laws of electrons and holes are derived from zero-, first-, and second-order moments of Boltzmann’s equations [4, 12]. In this section, we review the HD and the Poisson equations. The transport equations for electrons are given as follows:

$$\frac{\partial n}{\partial t} + \nabla \cdot (n \mathbf{u}_e) = \left[\frac{\partial n}{\partial t} \right]_{\text{col}}, \quad (1)$$

$$\frac{\partial \mathbf{p}_e}{\partial t} + \mathbf{u}_e (\nabla \cdot \mathbf{p}_e) + (\mathbf{p}_e \cdot \nabla) \mathbf{u}_e = -\epsilon n E - \nabla (n k_b T_e) + \left[\frac{\partial \mathbf{p}_e}{\partial t} \right]_{\text{col}}, \quad (2)$$

$$\frac{\partial w_e}{\partial t} + \nabla \cdot (\mathbf{u}_e w_e) = -\varepsilon n(\mathbf{u}_e \cdot \mathbf{E}) - \nabla \cdot (\mathbf{u}_e n k_b T_e) - \nabla \cdot \mathbf{q}_e + \left[\frac{\partial w_e}{\partial t} \right]_{\text{col}}. \quad (3)$$

Equations (1), (2) and (3) are the continuity equation and conservation laws for momentum and energy, respectively. A similar set of equations can be derived for holes:

$$\frac{\partial p}{\partial t} + \nabla \cdot (p \mathbf{u}_h) = \left[\frac{\partial p}{\partial t} \right]_{\text{col}}, \quad (4)$$

$$\frac{\partial \mathbf{p}_h}{\partial t} + \mathbf{u}_h (\nabla \cdot \mathbf{p}_h) + (\mathbf{p}_h \cdot \nabla) \mathbf{u}_h = \varepsilon p \mathbf{E} - \nabla (p k_b T_h) + \left[\frac{\partial \mathbf{p}_h}{\partial t} \right]_{\text{col}}, \quad (5)$$

$$\frac{\partial w_h}{\partial t} + \nabla \cdot (\mathbf{u}_h w_h) = \varepsilon p (\mathbf{u}_h \cdot \mathbf{E}) - \nabla \cdot (\mathbf{u}_h p k_b T_h) - \nabla \cdot \mathbf{q}_h + \left[\frac{\partial w_h}{\partial t} \right]_{\text{col}}. \quad (6)$$

The electron and hole concentrations are coupled to the electrostatic potential by the Poisson equation. The Poisson equation, derived from Maxwells equations [1, 13], is given by

$$\nabla \cdot (\theta \mathbf{E}) = \varepsilon (n - p - N_D^+ + N_A^-), \quad (7)$$

where the electric field \mathbf{E} is related to the electrostatic potential ψ by

$$\mathbf{E} = -\nabla \psi. \quad (8)$$

In (1)–(7), n and p are the concentration of electrons and holes; \mathbf{u}_e and \mathbf{u}_h are the electron and hole-velocity vectors; \mathbf{p}_e and \mathbf{p}_h are the electron and hole momentum density vectors; T_e and T_h are the electron and hole temperatures; w_e and w_h are the electron and hole energy densities; \mathbf{q}_e and \mathbf{q}_h are the electron and hole heat flux vectors; ε is the magnitude of an elementary charge; k_b is the Boltzmann constant; N_D^+ is the concentration of ionized donor and N_A^- the concentration of ionized acceptor; θ is the dielectric permittivity; $[\]_{\text{col}}$ denotes collision terms. Explicit form of collision terms for one-carrier devices are given in Section 2.3. In the above equations, vectors are denoted by bold letters.

The electron and hole conservation laws are coupled to the Poisson equation through the electric field term appearing on the right-hand side of the equations. Similarly, the Poisson equation is coupled to the electron and hole conservation laws through the concentrations of electrons and holes, which again appear on the right-hand side of the equation. This type of coupling can be considered ‘weak’ since the coupling terms act primarily as source terms. Due to the nonlinearity of the system, weak interaction between Poisson and hydrodynamic equations does not necessarily imply that the influence of the coupling on the solution is small. We discuss this issue of weak coupling and the solution strategy in more detail in Section 5.

Since the HD equations of electrons and holes are similar, the numerical treatment of the two systems is identical. In this paper, we focus on the formulation for the electron system. For clarity of presentation, the subscript $_e$ is removed in the sequel, it being understood that all variables missing a subscript pertain to the electron system.

The electron momentum and energy density can be written as

$$\mathbf{p} \doteq m n \mathbf{u}, \quad (9)$$

$$w = \frac{3}{2}nk_bT + \frac{1}{2}mn|\mathbf{u}|^2, \quad (10)$$

respectively, where m is the electron mass. The Fourier law for heat conduction is given by

$$\mathbf{q} = -\kappa n \nabla T, \quad (11)$$

where

$$\kappa = \frac{3\mu_{n0}k_b^2T_0}{2\varepsilon} \quad (12)$$

and where μ_{n0} is the electron mobility, and T_0 is the temperature of the lattice.

With appropriate modifications, the HD equations can be written to resemble the equations of compressible gas flow. More specifically, they take the form of Euler equations with $\gamma = 5/3$, for a gas of charged particles in an electric field with the addition of a heat conduction term. The momentum and energy conservation laws, noted in (2) and (3), respectively, can be simplified as shown in the following subsections.

2.1. Equation for conservation of momentum

Using indicial notation, equation (2) for the conservation of momentum, can be rewritten as

$$\frac{\partial}{\partial t}(mnu_i) + u_i(mnu_j)_{,j} + mnu_j(u_{i,j}) = -\varepsilon nE_i - (nk_bT)_{,i} + \left[\frac{\partial p_i}{\partial t}\right]_{\text{col}}, \quad (13)$$

where u_i , E_i and $[\partial p_i / \partial t]_{\text{col}}$ denote components of velocity, electric field, and collision terms. Repeated indices implies summation over a range of 1 to 3, and $(\cdot)_{,j}$ denotes differentiation with respect to the j th spatial coordinate. Dividing (13) by the electron mass m and simplifying, we obtain

$$\frac{\partial}{\partial t}(nu_i) + (nu_iu_j)_{,j} = -\frac{\varepsilon}{m}nE_i - \left(\frac{nk_bT}{m}\right)_{,i} + \frac{1}{m}\left[\frac{\partial p_i}{\partial t}\right]_{\text{col}}. \quad (14)$$

If we introduce the *electron pressure* per unit mass, defined as

$$P = \frac{nk_bT}{m}, \quad (15)$$

the momentum equation (14) can be written as

$$\frac{\partial}{\partial t}(nu_i) + (nu_iu_j + P\delta_{ij})_{,j} = -\frac{\varepsilon}{m}nE_i + \frac{1}{m}\left[\frac{\partial p_i}{\partial t}\right]_{\text{col}}, \quad (16)$$

where δ_{ij} is the Kronecker delta. Equation (16) is analogous to the Euler equation for conservation of momentum with the driving forces given in terms of the electric field and collision terms. The definition of electron pressure per unit mass arises naturally from this transformation procedure.

2.2. Equation for conservation of energy

Equation (3) for the conservation of energy can also be rewritten with indicial notation as follows:

$$\frac{\partial w}{\partial t} + (u_i w)_{,i} = -\epsilon n u_i E_i - (u_i n k_b T)_{,i} - q_{i,i} + \left[\frac{\partial w}{\partial t} \right]_{\text{col}}. \quad (17)$$

Introducing the term *energy density*, defined by

$$w = n m e_{\text{tot}}, \quad (18)$$

where e_{tot} denotes the total energy per unit mass, it follows that

$$e_{\text{tot}} = \frac{3}{2m} k_b T + \frac{1}{2} |\mathbf{u}|^2. \quad (19)$$

It is also useful to introduce the *electron internal energy* per unit mass defined as

$$e_{\text{int}} = \frac{3}{2m} k_b T. \quad (20)$$

The total energy per unit mass of an electron can be written as the sum of the internal energy and kinetic energy per unit mass, i.e.,

$$e_{\text{tot}} = e_{\text{int}} + \frac{1}{2} |\mathbf{u}|^2. \quad (21)$$

Using the above equations, the energy conservation equation (17) can be rewritten as

$$\frac{\partial}{\partial t} (n m e_{\text{tot}}) + (n m e_{\text{tot}} u_i + u_i n k_b T)_{,i} = -\epsilon n u_i E_i - q_{i,i} + \left[\frac{\partial w}{\partial t} \right]_{\text{col}}. \quad (22)$$

Substituting into (22) the electron pressure per unit mass, P , as defined in (15), we obtain

$$\frac{\partial}{\partial t} (n e_{\text{tot}}) + (n e_{\text{tot}} u_i + P u_i)_{,i} = -\frac{\epsilon n u_i E_i}{m} - q_{i,i} + \frac{1}{m} \left[\frac{\partial w}{\partial t} \right]_{\text{col}}, \quad (23)$$

where

$$q_i = \frac{-\kappa n T_{,i}}{m}. \quad (24)$$

Equation (23) is analogous to the Euler equation for conservation of energy with the driving forces expressed in terms of the electric field and collision terms, with the addition of a heat conduction term. Once again, the definitions of electron internal and total energies per unit mass arise naturally from this transformational procedure.

2.3. Collision terms

The collision terms $[\cdot]_{\text{col}}$ in (1), (2) and (3) describe the rate of change of mass, momentum and energy due to collisions. These terms account for the electron–electron and electron–lattice interactions, the energy transfer between electrons and lattice, and the generation and recombination processes. In the context of one-carrier devices, the case considered in this

paper, the explicit forms given below apply to a ballistic diode problem in which the effects of holes are neglected.

The collision term for the rate of change of mass is due to the generation and recombination processes. These processes are not present in single carrier devices and hence the collision term for the continuity equation is trivial, i.e.,

$$\left[\frac{\partial n}{\partial t} \right]_{\text{col}} = 0. \quad (25)$$

The collision terms in the momentum conservation, (2), and the energy conservation, (3), represent the rate of change of momentum and energy density, respectively, due to intraband collisions. These are expressed using momentum and energy relaxation times as [3, 6]

$$\left[\frac{\partial p}{\partial t} \right]_{\text{col}} = \frac{-p}{\tau_p}, \quad \left[\frac{\partial w}{\partial t} \right]_{\text{col}} = \frac{-(w - \frac{3}{2}nk_b T_0)}{\tau_w}, \quad (26)$$

where the momentum relaxation time is expressed as

$$\tau_p = m \frac{\mu_{n0}}{\epsilon} \frac{T_0}{T} \quad (27)$$

and the energy relaxation time is expressed as

$$\tau_w = \frac{3}{2} \frac{\mu_{n0}}{\epsilon v_s^2} \frac{k_b T T_0}{T + T_0} + \frac{\tau_p}{2} \quad (28)$$

and v_s is the saturation velocity.

2.4. Summary of HD equations for a semiconductor device

In summary, the modified set of HD equations for single carrier devices can be stated as follows:

$$\frac{\partial n}{\partial t} + (nu_i)_{,i} = 0, \quad (29)$$

$$\frac{\partial}{\partial t} (nu_i) + (nu_i u_j + P\delta_{ij})_{,j} = n \left[-\frac{\epsilon}{m} E_i - \frac{u_i}{\tau_p} \right], \quad (30)$$

$$\frac{\partial}{\partial t} (ne_{\text{tot}}) + (ne_{\text{tot}} u_i + Pu_i)_{,i} = -\frac{\epsilon n u_i E_i}{m} - q_{i,i} - \frac{1}{m} \frac{(nme_{\text{tot}} - \frac{3}{2}nk_b T_0)}{\tau_w}. \quad (31)$$

The HD equations are supplemented by constitutive relations, expressed in terms of thermodynamic quantities, as given below:

(i) The internal energy per unit mass, e_{int} , is defined as

$$e_{\text{int}} = c_v T, \quad c_v = \frac{3}{2m} k_b, \quad (32)$$

where c_v is the specific heat at constant volume.

(ii) The electron pressure per unit mass, P , can be expressed in terms of γ , the ratio of specific heats, as

$$P = (\gamma - 1)ne_{\text{int}}, \quad (33)$$

where

$$\gamma = \frac{c_p}{c_v} = \frac{5}{3} \quad \text{and} \quad c_p = \frac{5}{2m} k_b. \quad (34)$$

c_p is the specific heat at constant pressure. Equations (32) and (33) constitute the perfect gas law, i.e. they satisfy the relation $Pv = RT$, where $v = 1/n$ is the specific volume and $R = c_p - c_v$ is the specific gas constant.

2.5. Nondimensionalization of HD equations

In semiconductor devices, some of the physical quantities of interest are characterized by a very large range in magnitude. Thus, use of dimensional variables may result in ill-conditioning of the matrix problem to be solved. In addition, interpretation of the results may be difficult. This large differential of magnitudes among physical quantities can be addressed by nondimensionalizing the governing equations. Nondimensionalization can be performed on the set of equations (1)–(3) or on (29)–(31) since these two sets of equations are equivalent as shown in the previous sections. Here we discuss the nondimensionalization procedure based on the set (1)–(3).

The conservation laws as defined in (1), (2) and (3) can be made dimensionless if the dependent and independent variables are divided by certain constant reference properties. Some examples of reference properties are the velocity u_0 or the device length L . We select to nondimensionalize the variables as follows:

$$\begin{aligned} x_i^* &= \frac{x_i}{L}, & n^* &= \frac{n}{n_0}, & P^* &= \frac{P}{n_0 u_0^2}, & u_i^* &= \frac{u_i}{u_0}, \\ t^* &= \frac{tu_0}{L}, & \nabla^* &= L \nabla, & e_{\text{tot}}^* &= \frac{e_{\text{tot}}}{u_0^2}, \end{aligned} \quad (35)$$

where the dimensionless parameters are denoted by superscript asterisk. All other dimensional parameters are divided by a constant value of its own reference parameter. Using the above scalings, the continuity equation now takes the form

$$\frac{\partial}{\partial t^*} n^* + \nabla^* \cdot (n^* \mathbf{u}^*) = 0. \quad (36)$$

It is noted that the continuity equation has undergone a change of variables under these new transformations. We use a zero collision term for conservation of mass as discussed in Section 2.3. For the conservation of momentum, the nondimensionalized equation takes the form

$$\begin{aligned} \frac{\partial}{\partial t^*} \mathbf{p}^* + \mathbf{u}^* (\nabla^* \cdot \mathbf{p}^*) + (\mathbf{p}^* \cdot \nabla^*) \mathbf{u}^* \\ = -(\text{Ndp})_1 \varepsilon^* n^* \mathbf{E}^* - (\text{Ndp})_2 \nabla^* (n^* k_b^* T^*) + (\text{Ndp})_3 \left[\frac{\partial \mathbf{p}}{\partial t} \right]_{\text{col}}^*, \end{aligned} \quad (37)$$

where $(\text{Ndp})_1$, $(\text{Ndp})_2$ and $(\text{Ndp})_3$ are three nondimensional parameters defined as

$$(\text{Ndp})_1 = \frac{\varepsilon_0 E_0 L}{m_0 u_0^2}, \quad (\text{Ndp})_2 = \frac{k_{b0} T'_0}{m_0 u_0^2}, \quad (\text{Ndp})_3 = \frac{\varepsilon_0 L}{m_0 \mu'_{n0} u_0}. \quad (38)$$

In (38), ε_0 is the reference charge, m_0 is the reference mass, k_{b0} is the reference Boltzmann constant, T'_0 is the reference temperature and μ'_{n0} is the reference mobility.

For the energy equation, substituting the nondimensional parameters, we obtain

$$\begin{aligned} \frac{\partial}{\partial t^*} w^* + \nabla^* \cdot (u^* w^*) = & -(\text{Ndp})_1 \varepsilon^* n^* (u^* \cdot E^*) \\ & -(\text{Ndp})_2 \nabla^* \cdot (u^* n^* k_b^* T^*) - (\text{Ndp})_4 \nabla^* \cdot q^* + (\text{Ndp})_3 \left[\frac{\partial w}{\partial t} \right]_{\text{col}}^*, \end{aligned} \quad (39)$$

where

$$(\text{Ndp})_4 = \frac{\kappa_0 T'_0}{m_0 L u_0^3} \quad (40)$$

and κ_0 is the reference conductivity per unit volume. The four nondimensional parameters can be made unity by appropriate selection of reference quantities. In our work, the nondimensional coefficients are made unity by the following choice of reference values:

$$\begin{aligned} E_0 = \frac{V_t}{L}, \quad \varepsilon_0 = \frac{\theta_0 V_t}{n_0 L^2}, \quad k_{b0} = \frac{V_t \varepsilon_0}{T'_0}, \\ \mu'_{n0} = \frac{u_0 L}{V_t}, \quad m_0 = \frac{k_{b0} T'_0}{u_0^2}, \quad \kappa_0 = \frac{\mu'_{n0} k_{b0}^2 T'_0}{\varepsilon_0}. \end{aligned} \quad (41)$$

In the above equation, V_t is the reference voltage, θ_0 is the reference permittivity, and all other variables are as defined previously. It can be checked that all nondimensional coefficients are now unity.

In a similar manner, the Poisson equation can be transformed into a nondimensional form as

$$\nabla^* \cdot (\theta^* E^*) = \varepsilon^* (n^* - p^* - N_D^+ + N_A^-). \quad (42)$$

Equations (36), (37), (39) and (42) are the nondimensionalized set of device equations that are used in the finite element formulation. For the rest of this paper, we assume that the equations are dimensionless and the asterisk superscript is discarded from our notation for simplicity.

3. Conservation form and symmetrization

In this section we first give the conservation form of the HD equations, which is also known as the divergence law form. The conservation form leads to a quasi-linear system of equations

which involve unsymmetric matrix operators. For this reason we symmetrize the system of equations using entropy functions. Generalized entropy functions for compressible Euler and Navier–Stokes equations have been investigated by Harten [14]. These were enhanced in [10, 15, 16] to account for the heat conduction term. By following the ideas in these previous works, we symmetrize the HD equations. Variational formulations based on the symmetrized systems satisfy the second law of thermodynamics thereby establishing the stability of the solution. As shall be discussed in the next section, symmetrized systems provide the framework for the development of the Galerkin/least-squares method. Additional advantages include improving computational efficiency by employing a linear solver instead of a nonlinear solver, and global conservation under approximate element quadratures.

The HD equations can be written in conservation form as

$$U_{,i} + F_{i,i} = F_{i,i}^h + F, \quad (43)$$

where in three dimensions,

$$U = \begin{Bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \end{Bmatrix} = n \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{\text{tot}} \end{Bmatrix}, \quad (44)$$

$$F_i = nu_i \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{\text{tot}} \end{Bmatrix} + P \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_i \end{Bmatrix}, \quad (45)$$

$$F_i^h = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_i \end{Bmatrix} \quad (46)$$

and

$$F = \begin{Bmatrix} 0 \\ n \left[-\frac{\varepsilon}{m} E_1 - \frac{u_1 \varepsilon T}{m \mu_{n0} T_0} \right] \\ n \left[-\frac{\varepsilon}{m} E_2 - \frac{u_2 \varepsilon T}{m \mu_{n0} T_0} \right] \\ n \left[-\frac{\varepsilon}{m} E_3 - \frac{u_3 \varepsilon T}{m \mu_{n0} T_0} \right] \\ -\frac{\varepsilon n u_i E_i}{m} - \frac{1}{m} \frac{(n m e_{\text{tot}} - \frac{3}{2} n k_b T_0)}{\left[\frac{3}{2} \frac{\mu_{n0}}{\varepsilon v_s^2} \frac{k_b T T_0}{T + T_0} + \frac{\tau_p}{2} \right]} \end{Bmatrix}. \quad (47)$$

It is useful to rewrite the conservation form in the quasi-linear form

$$U_{,i} + A_i U_{,i} = (K_{ij} U_{,j})_{,i} + F, \quad (48)$$

where $A_i = F_{i,U}$ and $K_{ij} U_{,j} = F_i^h$. The matrices A_i do not possess the properties of symmetry or positiveness and, in general, are functions of U .

We seek a change of variables $U = U(V)$ to symmetrize the system given in (48) such that each of the coefficient matrices is symmetric. This can be achieved by considering a generalized scalar valued entropy function of the form $\mathcal{H} = \mathcal{H}(U) = -ns$, where s is the thermodynamic entropy per unit mass. We introduce a change of variables $U \rightarrow V$ defined by

$$V^i = \frac{\partial \mathcal{H}}{\partial U_i}, \quad (49)$$

V is referred to as the vector of (physical) entropy variables. In particular, the system is symmetrized by taking

$$s = c_v \ln \left(\frac{P}{P_0} \left(\frac{n}{n_0} \right)^{-\gamma} \right) + s_0, \quad (50)$$

where s_0 is the reference entropy, n_0 and P_0 are reference concentration and pressure, respectively. The new variables V^i are computed by using the chain rule

$$V^i = \mathcal{H}_{,U_i} = \mathcal{H}_{,Y} (U_{,Y})^{-1}, \quad (51)$$

where

$$Y = \begin{Bmatrix} v \\ u \\ e_{\text{int}} \end{Bmatrix} \quad (52)$$

and v is the specific volume. Using the definition of Y , we obtain

$$\mathcal{H}_{,Y} = \begin{bmatrix} \frac{s-R}{v^2} & 0 & 0 & 0 & -\frac{1}{vT} \end{bmatrix} \quad (53)$$

and

$$U_{,Y} = \begin{bmatrix} -\frac{1}{v^2} & 0 & 0 & 0 & 0 \\ -\frac{u_1}{v^2} & \frac{1}{v} & 0 & 0 & 0 \\ -\frac{u_2}{v^2} & 0 & \frac{1}{v} & 0 & 0 \\ -\frac{u_3}{v^2} & 0 & 0 & \frac{1}{v} & 0 \\ -\frac{e_{\text{tot}}}{v^2} & \frac{u_1}{v} & \frac{u_2}{v} & \frac{u_3}{v} & \frac{1}{v} \end{bmatrix}. \quad (54)$$

The new entropy variables are thus obtained from (51) as

$$V = \frac{1}{T} \begin{Bmatrix} \mu - \frac{1}{2} |u|^2 \\ u_1 \\ u_2 \\ u_3 \\ -1 \end{Bmatrix}, \quad (55)$$

where $\mu = e_{\text{int}} + Pv - Ts$ is the specific chemical potential.

Using the change of variables, the system of equations given in (48) can be rewritten as

$$\tilde{A}_0 V_{,i} + \tilde{A}_i V_{,i} = (\tilde{K}_{ij} V_{,j})_{,i} + F, \quad (56)$$

where

$$\tilde{A}_0 = U_{,v}, \quad \tilde{A}_i = A_i \tilde{A}_0, \quad \tilde{K}_{ij} = K_{ij} \tilde{A}_0. \quad (57)$$

In the above definitions, \tilde{A}_0 is symmetric and positive definite and \tilde{A}_i is symmetric. The explicit definitions of all the coefficient matrices are summarized in Appendix A. These coefficient matrices are first given in [16] for the compressible Euler and Navier–Stokes equations.

In addition to the matrices defined above, it is useful to express the source vector as a product of a coefficient matrix \tilde{C} and the vector V :

$$F = -\tilde{C}V. \quad (58)$$

The definition of \tilde{C} is not unique. In Appendix A, we have included one possible definition of \tilde{C} which is symmetric and positive definite.

4. Finite element formulation

This section presents a finite element formulation for the HD equations and the Poisson equation. For the HD equations, we enhance the space-time finite element formulations developed for compressible Navier-Stokes equations to account for the highly nonlinear source terms. The standard Galerkin finite element method is employed for the Poisson equation.

4.1. Finite element method for HD equations

The standard Galerkin finite element method exhibits spurious oscillations and poor stability properties for advective–diffusive systems in which the exact solution may be nonsmooth or discontinuous [8]. This deficiency led to the development of the Streamline-Upwind/Petrov–Galerkin (SUPG) method, which exhibits good stability properties and higher order accuracy [9, 17, 18]. The essential idea in the SUPG method is the addition of stabilizing terms, which introduces artificial diffusion in the Galerkin method to provide control over the advective derivative term. Since SUPG is a higher order linear method, monotone approximations of sharp layers is not possible. Thus some undershoot and/or overshoot may appear in the solution. Nonlinear shock capturing operators have been developed to overcome these undershoot and/or overshoot problems [10, 19, 20].

Galerkin/least-squares finite element methods are simple extensions to SUPG methods [21]. The methods coincide with SUPG methods in the absence of diffusion and source terms, and provide a more general framework than SUPG methods in the presence of diffusion and source terms. Terms of a least-squares type are added to the Galerkin method to obtain stability. The least-squares terms vanish at the exact solution thus establishing consistency.

The temporal behavior of the problem is accounted for by using a discontinuous in time Galerkin approximation [25]. In the space-time Galerkin/least-squares method, the solution is obtained by marching sequentially through time; the solution of the system of equations at each time step is computed based on the solution obtained at the previous time step. In the following we develop the variational equation and then the finite element discretization for steady state problems.

4.1.1. Variational formulation

Let $0 = t_0 < t_1 < \dots < t_N = T$ be a sequence of time levels and $Q_n = \Omega \times I_n$ be a sequence of time-slabs in which Ω is the spatial domain and $I_n = (t_n, t_{n+1})$ is a time interval. Let $(n_{e1})_n$ denote the number of space-time elements in Q_n , and $Q_n^e = \Omega_n^e \times I_n$ denote the space time element domain in the n th time slab with Ω_n^e the discretization of the spatial domain in the n th time slab. The space of trial functions is

$$\mathcal{S}_n^h = \{V^h \mid V^h \in H^1(Q_n), D(V^h) = g(t) \text{ on } B_n\}, \quad (59)$$

where $B_n = \Gamma \times I_n$ denotes the boundary of the n th space time slab, D is the nonlinear boundary operator, and g is the prescribed boundary condition. The space of weighting functions is

$$\mathcal{V}_n^h = \{W^h \mid W^h \in H^1(Q_n), D'(W^h) = 0 \text{ on } B_n\}, \quad (60)$$

where D' is the nonlinear boundary condition operator.

Before stating the variational equation, it is useful to introduce the following notation:

$$(W^h, V^h)_{Q_n} = \int_{Q_n} (W^h \cdot V^h) dQ, \quad (61)$$

$$(W^h, V^h)_{\Omega} = \int_{\Omega} (W^h \cdot V^h) d\Omega, \quad (62)$$

$$a(W^h, V^h)_{Q_n} = \int_{Q_n} (W_i^h \cdot \tilde{K}_{ij} V_j^h) dQ, \quad (63)$$

$$(W^h, V^h)_{B_n} = \int_{B_n} (W^h \cdot V^h) n_i dB, \quad (64)$$

$$(W^h, V^h)_{Q_n^{\Sigma}} = \sum_{e=1}^{(n_{e1})_n} \int_{Q_n^e} (W^h \cdot V^h) dQ. \quad (65)$$

The space-time Galerkin/least-squares formulation for the symmetrized electron system equation (56) can be stated as follows. Within each Q_n , $n = 0, \dots, n-1$, find $V^h \in \mathcal{S}_n^h$ such

that for all $W^h \in \mathcal{V}_n^h$ the following variational equation is satisfied:

$$B_{\text{GLS}}(W^h, V^h)_n = L_{\text{GLS}}(W^h)_n, \quad (66)$$

where

$$B_{\text{GLS}}(W^h, V^h)_n = B(W^h, V^h)_n + (\mathcal{L}W^h, \tau \mathcal{L}V^h)_{Q_n^\Sigma} + B_{\text{DC}}(W^h, V^h), \quad (67)$$

$$\begin{aligned} B(W^h, V^h)_n = & (-W^h_t, U(V^h))_{Q_n} + (W^h_t, F_i(V^h))_{Q_n} + a(W^h, V^h)_{Q_n} + (W^h, F(V^h))_{Q_n} \\ & + (W^h(t_{n+1}^-), U(V^h(t_{n+1}^-)))_{\Omega} + (W^h, F_i(V^h) - F_i^h(V^h))_{B_n}, \end{aligned} \quad (68)$$

$$B_{\text{DC}}(W^h, V^h) + (v^h \hat{\nabla}_\xi W^h, [[\hat{A}_0]] \hat{\nabla}_\xi V^h)_{Q_n^\Sigma}, \quad (69)$$

$$L_{\text{GLS}}(W^h)_n = L(W^h)_n = (W^h(t_n^+), U(V^h(t_n^-)))_{\Omega}. \quad (70)$$

With regard to (67)–(70), the following remarks are applicable:

(i) The first term on the right-hand-side of (67) constitutes the time-discontinuous Galerkin formulation, which is given in (68).

(ii) The second integral product in (67) is the least-squares operator which is nonlinear in both W^h and V^h . The symmetric positive semidefinite, $n_{\text{dof}} \times n_{\text{dof}}$ matrix τ contains Galerkin/least-squares parameters whose selection is discussed in [22]. τ can be interpreted as a matrix of intrinsic time scales. The number of degrees of freedom of the problem are n_{dof} , and \mathcal{L} is the governing differential operator of the problem defined from (56) as

$$\mathcal{L} = \tilde{A}_0 \frac{\partial}{\partial t} + \tilde{A}_i \frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i} \left(\tilde{K}_{ij} \frac{\partial}{\partial x_j} \right) + \tilde{C}. \quad (71)$$

(iii) The third term in (67) is a discontinuity-capturing operator and is also nonlinear in both W^h and V^h . The integral product definition of this term is given in (69). $\hat{\nabla}_\xi$ is defined as the generalized local coordinates gradient operator. v^h is a scalar discontinuity-capturing factor having the dimension of reciprocal of time, and

$$[[\tilde{A}_0]] = \begin{bmatrix} \tilde{A}_0 & & & \\ & \tilde{A}_0 & & \\ & & \ddots & \\ & & & \tilde{A}_0 \end{bmatrix}. \quad (72)$$

The selection of v^h has been discussed in [10].

(iv) Equation (70) is the contribution of the jump condition term. Jump condition is added to the variational form to enforce weak initial conditions for each space–time slab, and introduce numerical dissipation. The jump condition is given by

$$\int_{\Omega} W^h(t_n^+) \cdot [[U(V^h(t_n))]] \, d\Omega, \quad (73)$$

where

$$[[U(t_n)]] = U(t_n^+) - U(t_n^-) \quad (74)$$

denotes the jump in time of U .

4.1.2. Finite element discretization

A computationally efficient scheme for steady state problems can be developed by considering the finite element spaces to be constant in time within each space–time slab and discontinuous across the space–time slab interfaces. Within the n th space–time slab, the finite element trial solution and the weighting function are taken to be

$$\mathbf{V}^h = \sum_{A=1}^{(n_{np})_n} N_A^{(n)}(\mathbf{x}) \mathbf{v}_{A:(n+1)}, \quad \mathbf{W}^h = \sum_{A=1}^{(n_{np})_n} N_A^{(n)}(\mathbf{x}) \mathbf{w}_{A:(n+1)}, \quad \text{for } \mathbf{x} \in \Omega, \quad (75)$$

where $\mathbf{v}_{A:(n+1)}$ and $\mathbf{w}_{A:(n+1)}$ are, respectively, the $n_{\text{dof}} \times 1$ vectors of nodal unknowns and weighting functions at node A for the n th space–time slab. $(n_{np})_n$ is the number of nodal points for the n th space–time slab, and $N_A^{(n)}(\mathbf{x})$ is the finite element spatial shape-function of node A for the n th space–time slab (the subscripts and superscripts are dropped from now on to simplify the notation). Defining

$$\mathbf{v} = \{\mathbf{v}_A^t\}^t, \quad \mathbf{w} = \{\mathbf{w}_A^t\}^t, \quad \mathbf{v}_n = \{\mathbf{v}_{A:n}^t\}^t, \\ A = 1, \dots, n_{np}, \quad (76)$$

and substituting the finite element approximations (75), into the space–time Galerkin/least-squares variational equation (66), we obtain

$$\mathbf{w} \cdot \mathbf{G}(\mathbf{v}; \mathbf{v}_{(n)}) = 0, \quad (77)$$

where $\mathbf{G}(\mathbf{v}; \mathbf{v}_{(n)})$ is a system of nonlinear algebraic equations with an unknown vector \mathbf{v} . Since (77) must hold for all unconstrained coefficients \mathbf{w} , it follows that

$$\mathbf{G}(\mathbf{v}; \mathbf{v}_{(n)}) = 0. \quad (78)$$

Equation (78) is the nonlinear finite element matrix equation in which there are $n_{np} \times n_{\text{dof}}$ equations and $n_{np} \times n_{\text{dof}}$ unknowns.

The nonlinear system can be linearized with respect to the unknown vector \mathbf{v} , and a time stepping solution algorithm can be employed in the format of the predictor multi-corrector algorithm. At each time slab n , if we denote $\mathbf{v}^{(i)}$ to be the i th iterative approximation of $\mathbf{v}_{(n+1)}$, with $\mathbf{v}^{(0)} = \mathbf{v}_{(n)}$, linearization of (78) gives

$$\mathbf{R}^{(i)} + \mathbf{M}^{(i)} \Delta \mathbf{v}^{(i)} = 0, \quad (79)$$

where

$$\Delta \mathbf{v}^{(i)} = \mathbf{v}^{(i)} - \mathbf{v}^{(i-1)}. \quad (80)$$

$\mathbf{R}^{(i)}$ and $\mathbf{M}^{(i)}$ denote the residual vector and the consistent tangent matrix at the i th iteration, respectively. The predictor multi-corrector algorithm can now be summarized as follows:

*For each time step, n , do
begin*

```

Predictor:  $\mathbf{v}^{(0)} = \mathbf{v}_{(n)}$  ;
For each corrector  $i = 0, 1, \dots, n_{\text{cor}} - 1$  do
begin /* corrector loop */
    solve  $\mathbf{M}^{(i)} \Delta \mathbf{v}^{(i)} = -\mathbf{R}^{(i)}$  ;
     $\mathbf{v}^{(i+1)} = \mathbf{v}^{(i)} + \Delta \mathbf{v}^{(i)}$  ;
end .
 $\mathbf{v}_{(n+1)} = \mathbf{v}^{(n_{\text{cor}})}$  ;
end .

```

In the above procedure, n_{cor} denotes the number of correctors.

4.2. Finite element model for the Poisson equation

Finite element formulation of the Poisson equation given in (7) is rather straightforward and is briefly summarized in this section. The space of the trial functions is

$$\mathcal{S}_1^h = \{ \psi^h \mid \psi^h \in H^1(\Omega), \psi^h = g_p \text{ on } \Gamma_g \} \quad (81)$$

and the space of weighting functions is

$$\mathcal{V}_1^h = \{ \bar{\psi}^h \mid \bar{\psi}^h \in H^1(\Omega), \bar{\psi}^h = 0 \text{ on } \Gamma_g \}, \quad (82)$$

where g_p are the prescribed essential boundary conditions applied at the boundary Γ_g .

The weak form of the problem can be stated as follows: Find $\psi^h \in \mathcal{S}_1^h$ such that for all $\bar{\psi}^h \in \mathcal{V}_1^h$, the following equation is satisfied:

$$\int_{\Omega} \left(\frac{\partial \bar{\psi}^h}{\partial x_i} \frac{\partial \psi^h}{\partial x_i} + \bar{\psi}^h \varepsilon (n - p - N_D^+ + N_A^-) \right) d\Omega - \int_{\Gamma_h} \bar{\psi}^h h_i d\Gamma = 0, \quad (83)$$

where $(\partial \psi^h / \partial x_i) k_i = h_i$ are the natural boundary conditions prescribed on the portion of the boundary Γ_h , and k_i denotes the unit outward normal to the boundary Γ_h .

Using standard finite element discretization [23], a matrix form is obtained which is solved for the potential ψ^h at all nodes. The electric fields are computed at the center of each element and then projected onto the mesh nodes using smoothing procedures of a least-squares type [24].

5. Solution schemes

This section discusses an algorithmic approach for solving coupled HD and Poisson equations. One approach is to solve the coupled HD and Poisson problems simultaneously. However, for the one-carrier devices that we consider in this paper, the coupling between the electron HD equations and the Poisson equation is through the source terms. The collision terms presented in Section 2.3 do not couple with the Poisson equation. A staggered scheme appears attractive for this weakly coupled system of Poisson and HD equations. Computation-

ally, the staggered scheme that treats the Poisson equation and the HD equations separately is more efficient than solving both equations as a single system.

In the staggered scheme, we first solve the Poisson equation for the potential and electric fields. We use the computed electric fields and solve the HD equations for concentrations, velocities and temperature. The computed concentrations are then taken as input for the Poisson equation to calculate the electric fields. This iterative procedure is stopped when all the equations are satisfied within a given tolerance parameter for convergence. Although we have not pursued mathematical proofs for stability of the staggered scheme, our experience on the test examples presented in the next section indicate that this solution scheme is quite stable.

The solution is said to reach a steady state when the residual is constant and does not decrease any further. The constant in time approximation for finite element spaces provides a very attractive time marching scheme for steady state problems. This scheme, however, provides low order of accuracy in time and may not be considered sufficiently accurate for transient problems. For transient problems, high order of accuracy in time can be provided by employing linear in time finite element spaces; this subject is beyond the scope of this paper.

6. Numerical results

The numerical algorithms presented in the previous sections are tested for one- and two-dimensional single-carrier devices. This section describes the results obtained to illustrate the applicability of the finite element formulation for semiconductor device problems. First, we will treat a traditional example, an $n^+ - n - n^+$ silicon diode, to verify the results of our code against those reported in literature. Next, we will discuss a simple extension of this problem to a 2-D problem. The intention is to show the generality of our approach; no modifications in our formulation need to be made to deal with 2-D and/or 3-D problems. In a future paper, we will report on results for more complex and more interesting devices. Here, we focus on the numerical capabilities of the proposed finite element formulation.

6.1. Example 1: 1-D problem

Computational experiments are performed on a $0.6\text{-}\mu\text{m}$ $n^+ - n - n^+$ silicon diode at 300 K with $n^+ = 5.0 \times 10^{17} \text{ cm}^{-3}$ and $n = 2.0 \times 10^{15} \text{ cm}^{-3}$. The doping in the $n^+ - n$ transition region varies as a Gaussian function with a $\sigma = 0.01 \mu\text{m}$, the length of the n -region is approximately $0.4 \mu\text{m}$. The boundary conditions applied are given as follows:

$$\text{at } x = 0 \mu\text{m}, \quad n = 5.0 \times 10^{17} \text{ cm}^{-3}, \quad T = T_0 = 300 \text{ K}, \quad \psi = \psi_b(n_d);$$

$$\text{at } x = 0.6 \mu\text{m}, \quad n = 5.0 \times 10^{17} \text{ cm}^{-3}, \quad T = T_0 = 300 \text{ K}, \quad \psi = \psi_b(n_d) + \psi_{\text{appl}}.$$

ψ_b is the built-in potential defined as

$$\psi_b = \frac{k_b T}{\epsilon} \ln \left(\frac{n_d}{n_i} \right),$$

where n_d is the doping and n_i is the intrinsic concentration. ψ_{appl} denotes the applied bias which is taken as 1.5 V or 2.0 V (we show numerical results for both cases). The initial conditions for the time-marching scheme that we employ to reach steady state are as follows:

$$\text{at } t = 0, \quad n(x, 0) = n_d(x), \quad u(x, 0) = 0.0, \quad T(x, 0) = T_0.$$

In these results, no continuation method is used, i.e. the bias is applied in a single load step. We used 101 mesh points for this problem and with this mesh size, the least-squares terms are sufficient to smooth the solution near discontinuities, i.e. we do not need to use shock capturing operators.

The steady state results for this problem are shown in Figs. 1–5. Our results agree very well with the results previously reported in the literature [2, 5–7]. It should be noted here that, although the physical ‘truth’ of these solutions is still being debated, the numerical results obtained in this study prove the accuracy of our formulation and therefore support the notion that this formulation can very well be used to investigate exactly what would be the right physical model formulation.

6.2. Example 2: 2-D problem

Our two-dimensional example is a simple extension of the one-dimensional problem discussed above. The geometry of the device is shown in Fig. 6. The dark lines indicate the contact positions. Contacts 4–5 and 4–6 are terminated at a distance of $0.07 \mu\text{m}$ from the top left corner. The doping profile is given by

$$n_d(x, y) = 5.0 \times 10^{17} \text{ cm}^{-3}, \quad \text{for } 0.0 \leq x \leq 0.6 \text{ and } 0.0 \leq y \leq 0.1,$$

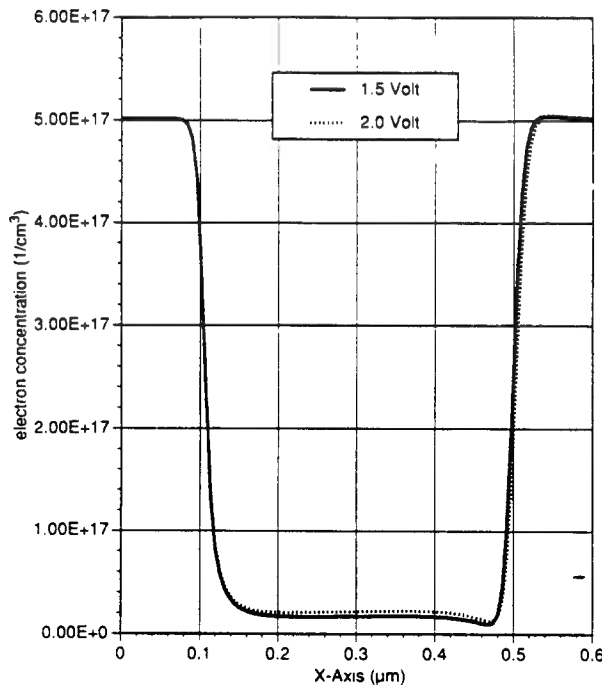


Fig. 1. Electron concentration (cm^{-3}) in steady state.

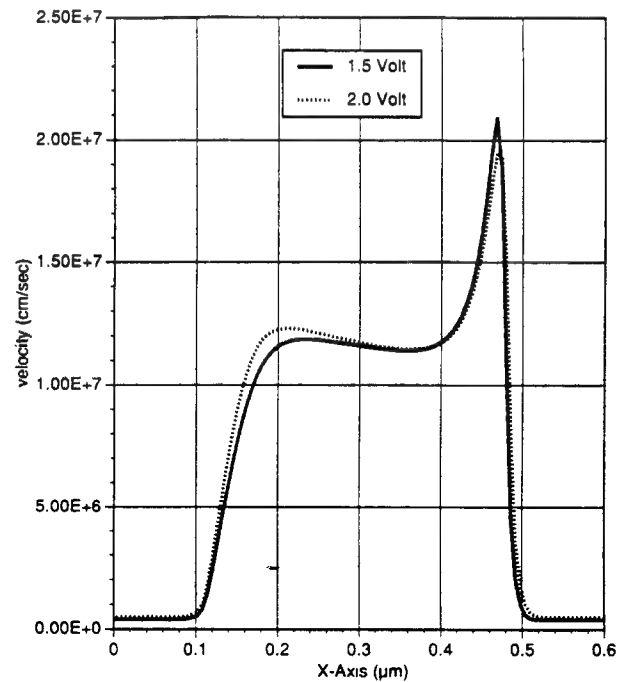


Fig. 2. Electron velocity (cm/s) in steady state.

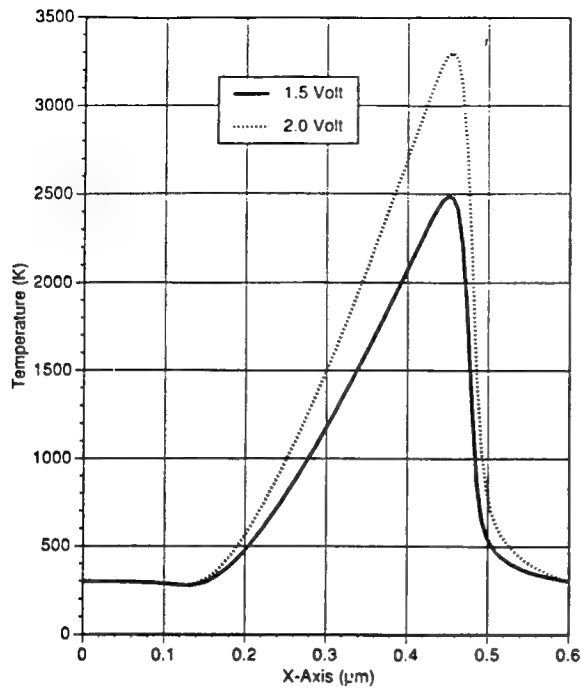


Fig. 3. Temperature (K) in steady state.

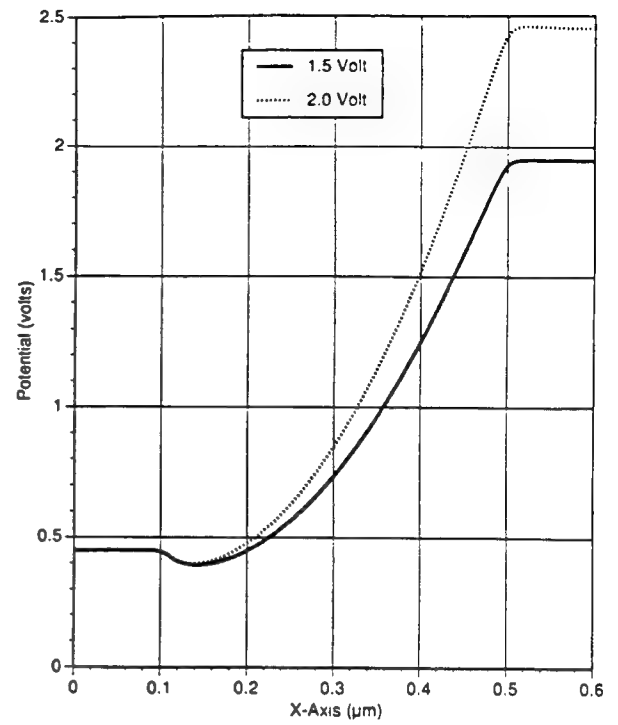


Fig. 4. Electrostatic potential (V) in steady state.

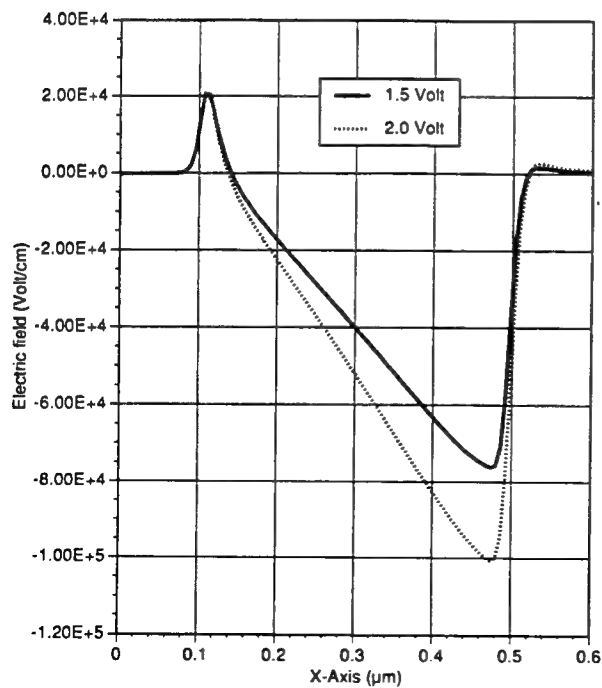
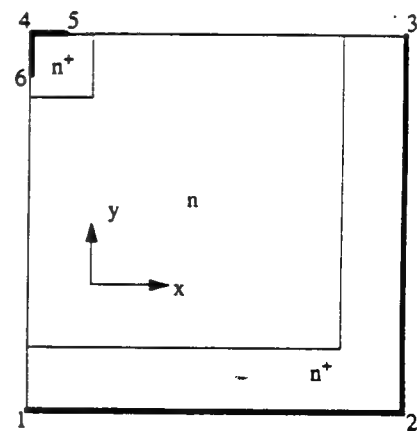


Fig. 5. Electric field (V/cm) in steady state.

Fig. 6. A $0.6 \mu\text{m} \times 0.6 \mu\text{m}$ $n^+ - n - n^+$ silicon device. Contacts are denoted by dark lines.

$$n_d(x, y) = 5.0 \times 10^{17} \text{ cm}^{-3}, \quad \text{for } 0.5 \leq x \leq 0.6 \text{ and } 0.2 \leq y \leq 0.6,$$

$$n_d(x, y) = 5.0 \times 10^{17} \text{ cm}^{-3}, \quad \text{for } 0.0 \leq x \leq 0.1 \text{ and } 0.5 \leq y \leq 0.6,$$

$$n_d(x, y) = 2.0 \times 10^{15} \text{ cm}^{-3}, \quad \text{elsewhere, with abrupt junctions.}$$

The boundary conditions we used for this problem are summarized as follows:

- (i) along contact 1–2: $n(x, 0) = n_d(x, 0)$, $u(x, 0) = 0.0$, $T(x, 0) = T_0 = 300 \text{ K}$, and $\psi(x, 0) = \psi_b + 1.0 \text{ V}$.
- (ii) along contact 2–3: $n(0.6, y) = n_d(0.6, y)$, $v(0.6, y) = 0.0$, $T(0.6, y) = 300 \text{ K}$, and $\psi(0.6, y) = \psi_b + 1.0 \text{ V}$.
- (iii) along contact 4–5: $n(x, 0.6) = n_d(x, 0.6)$, $u(x, 0.6) = 0.0$, $T(x, 0.6) = 300 \text{ K}$, and $\psi(x, 0.6) = \psi_b$.
- (iv) along contact 4–6: $n(0, y) = n_d(0, y)$, $v(0, y) = 0.0$, $T(0, y) = 300 \text{ K}$, and $\psi(0, y) = \psi_b + 1.0 \text{ V}$.
- (v) along boundary 5–3: $v = 0.0$, and Neumann boundary conditions for temperature and potential.
- (vi) along boundary 6–1: $u = 0.0$ and Neumann boundary conditions for temperature and potential.

The initial conditions are taken as $n(x, y) = n_d(x, y)$, $u(x, y) = v(x, y) = 0.0$, and $T(x, y) = T_0 = 300 \text{ K}$. We use a relatively coarse grid of 61×61 mesh points. The steady-state results for this problem are shown in Figs. 7–13. In order to simulate a realistic device, contacts are not extended to the full n^+ region near the top left corner as shown in Fig. 6.

In Fig. 8, the horizontal velocity u obtained at the steady state is shown. As expected, the solution along the line $y = 0.6 \mu\text{m}$ is very similar to the 1-D case. The global pattern of the solution can easily be understood from the 2-D character of the problem. There are two small

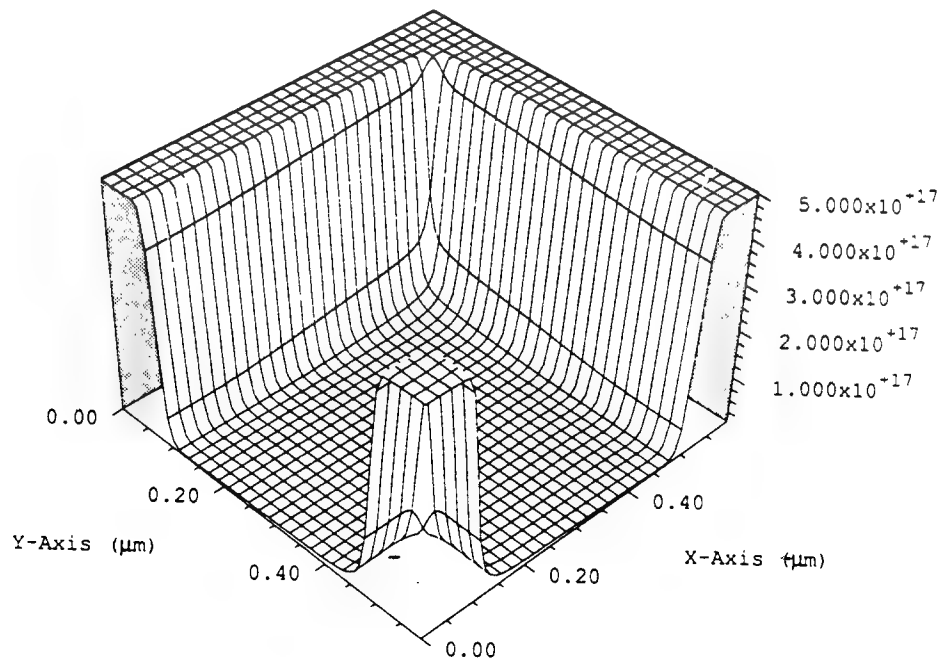


Fig. 7. Electron concentration (cm^{-3}) in steady state.

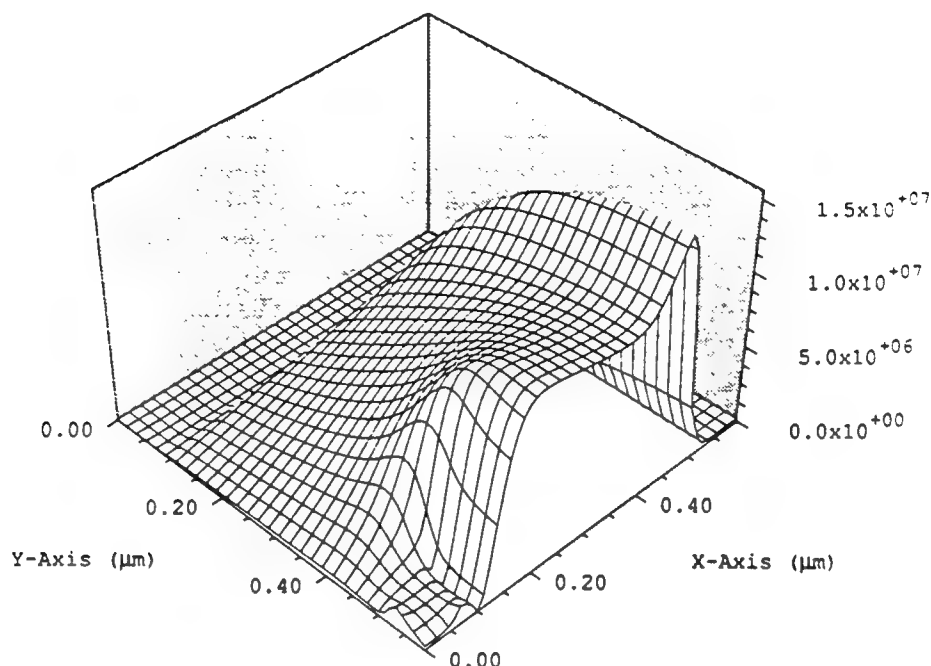


Fig. 8. Horizontal velocity (cm/s) in steady state.

details in the solution that need explanation, namely the small peaks in the velocity very close to the front corner of the device along both axes, between the edge of the contact and the boundary of the n^+ doping region. The peak along the x -axis is field driven. The electrons come out of the contact along the x -axis with in essence only a velocity in the y -direction. Those electrons entering the device very close to the end of the x -axis contact are immediately

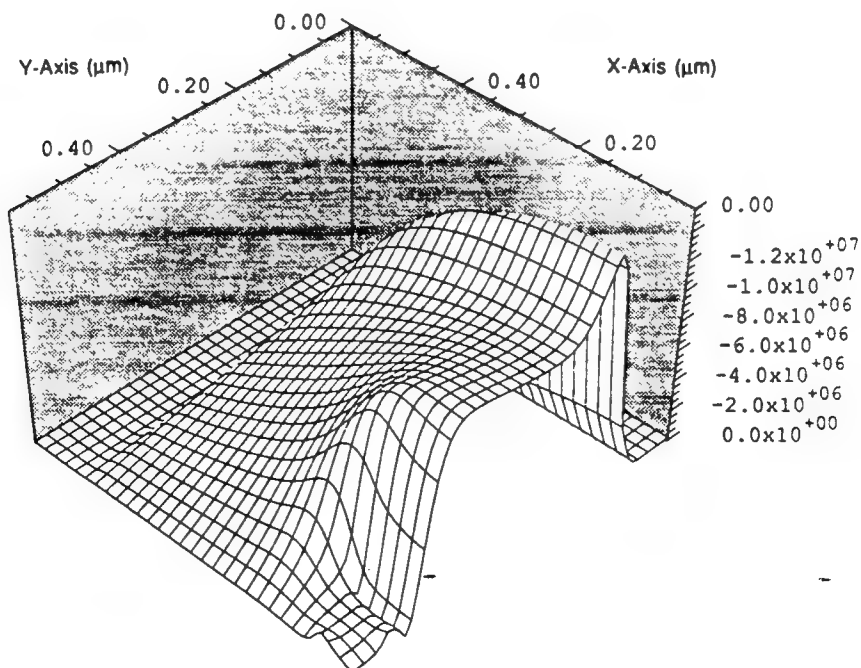


Fig. 9. Vertical velocity (cm/s) in steady state.

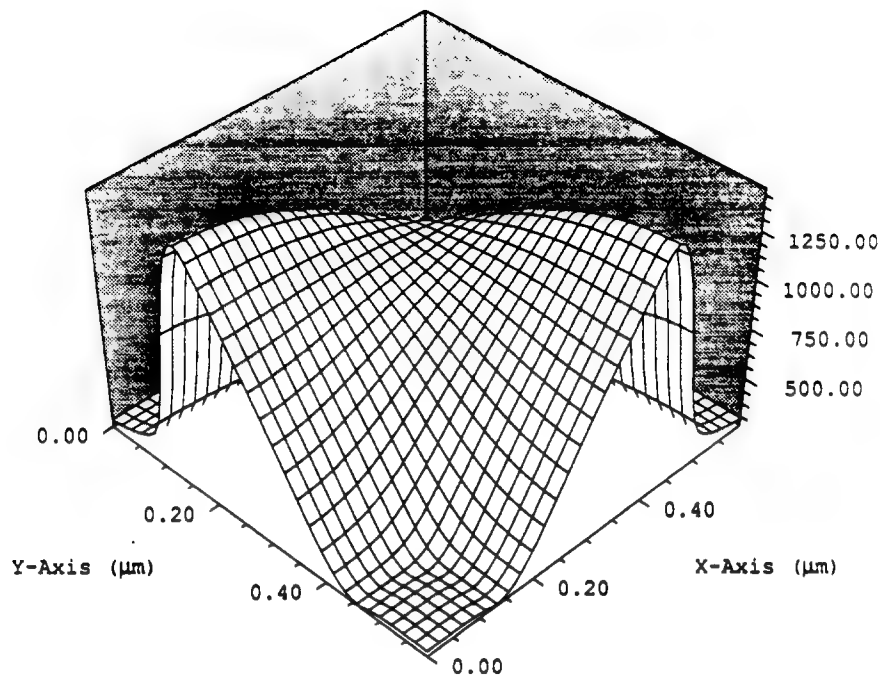


Fig. 10. Temperature (K) in steady state.

accelerated into the x -direction by the built-in electrical field in the diode junction. This explains the little bump in the x -velocity to the right of the x -axis contact. The peak along the y -axis has a different origin. Here, we are dealing with electrons streaming out of the y -axis contact and therefore with a tendency to pick up good x -velocity. The electrons coming out of this contact very close to the corner are, however, hampered in picking up speed because they

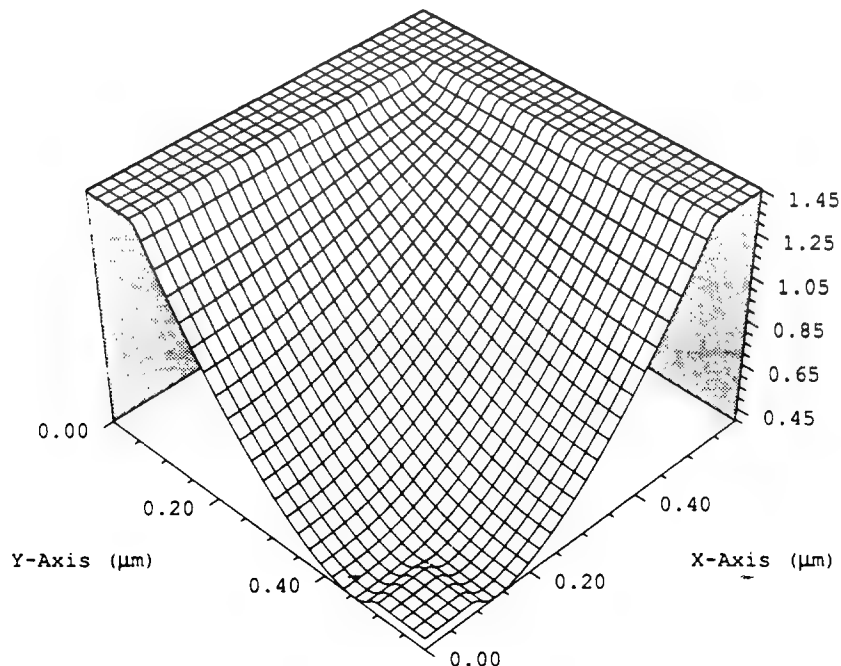


Fig. 11. Electrostatic potential (V) in steady state.

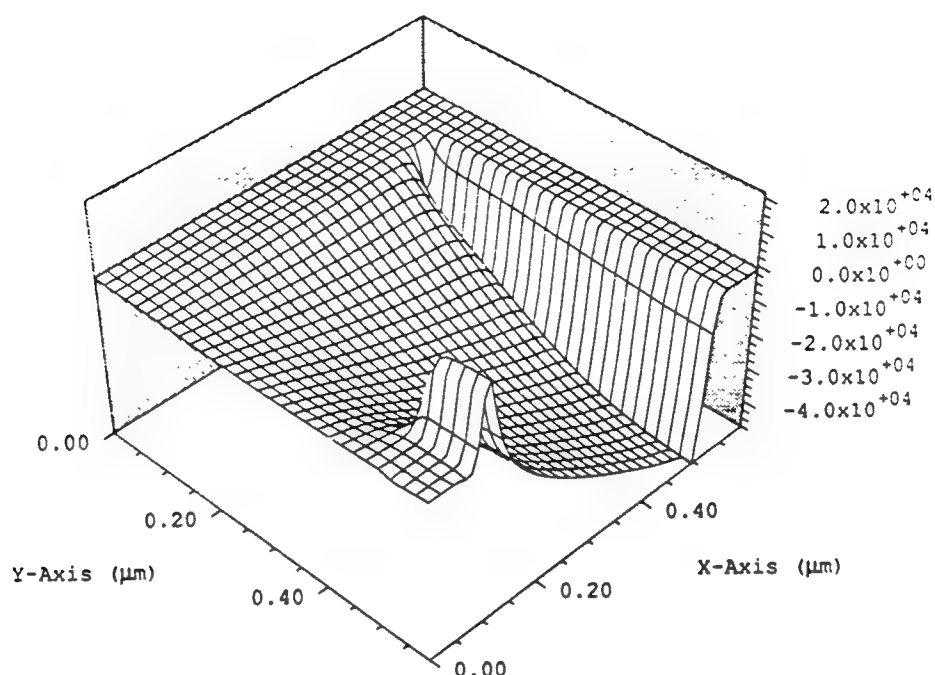


Fig. 12. Horizontal component of electric field (V/cm) in steady state.

'collide' with the electrons coming out of the x -axis contact close to the corner. The further we go along the y -axis away from the corner, the less this 'hampering' effect becomes as evidenced by the corresponding increase in x -velocity. Figure 9 shows the y -velocity. As expected from the symmetry of the problem, the profile is as good as identical to the one for x -velocity. Also the temperature data relate very well to the 1-D solutions and therefore are

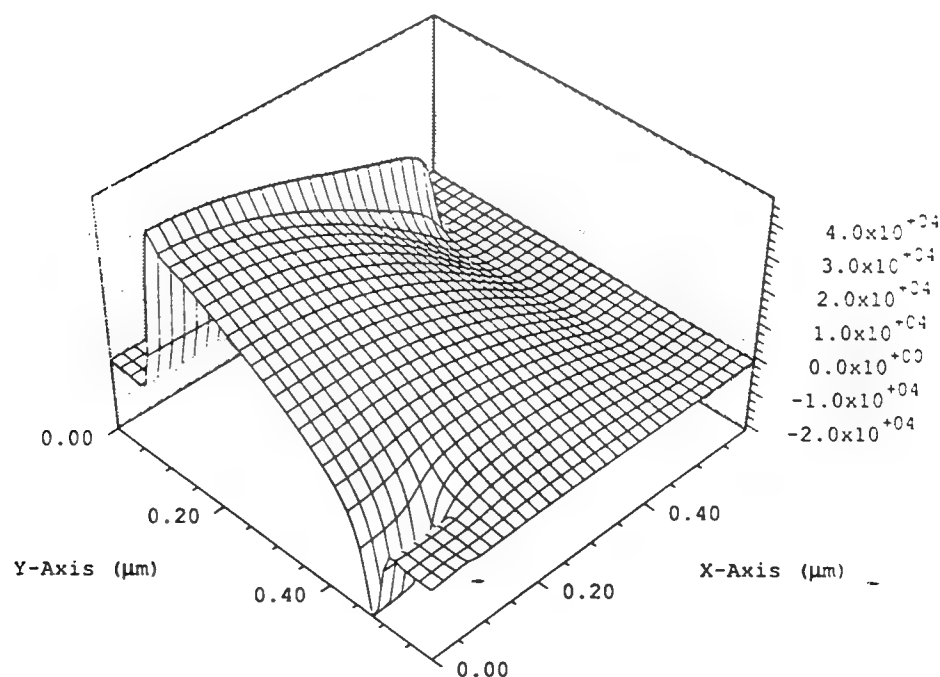


Fig. 13. Vertical component of electric field (V/cm) in steady state.

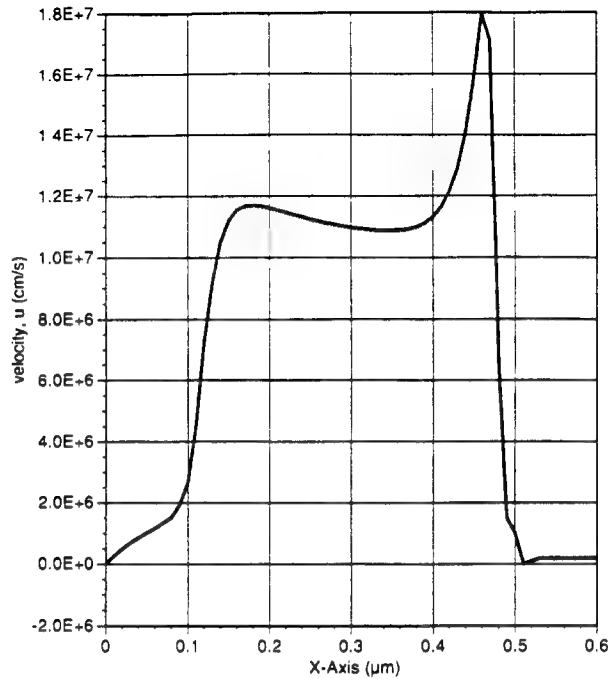


Fig. 14. Horizontal component of velocity along $y = 0.490$ without shock capturing operator.

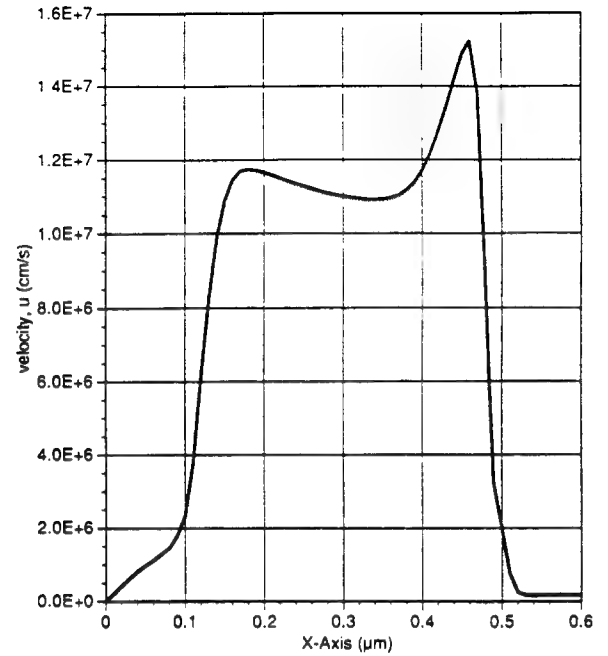


Fig. 15. Vertical component of velocity along $y = 0.490$ with shock capturing operator.

assumed to be accurate. Once again, we did not need to use the continuation method and the entire data bias specified is applied in a single step. However, in this example we need to use the shock capturing operators to eliminate small undershoots and overshoots given the coarseness of the mesh. In Figs. 14 and 15, horizontal component of velocity along $y = 0.490$ with and without shock capturing operators, respectively, are shown.

7. Summary

In this paper a general space–time Galerkin/least-squares finite element formulation for solving the HD equations of semiconductor devices is presented. Nonlinear shock capturing operators developed in the context of fluid flow problems have been enhanced to accommodate the highly nonlinear source terms present in the HD model, and were found to be useful to eliminate undershoots and overshoots near discontinuities. Numerical results reveal velocity overshoot, consistent with previously reported data. It is interesting to note that the heat flux term plays an important role in the simulation of velocity overshoot. When the heat flux term is neglected, unreasonable results with no velocity overshoot are observed. Well posed boundary conditions for 2D and 3D hydrodynamic models for semiconductor device problems are not clearly understood, contrary to the situation for compressible Euler and Navier–Stokes equations. This can be attributed to the need for a velocity boundary condition at a contact, which seems unphysical for device simulation. In our numerical studies, we found our algorithms to be stable even when we did not specify mathematically adequate boundary conditions. Specification of well posed, and physical boundary conditions is an area that requires further investigation.

The method described in this paper for semiconductor device equations is computationally very expensive. Current and future work will involve parallelizing the finite element software on multi-processing architecture, solving two-carrier devices in 2D as well as 3D, and developing an adaptive version of the finite element method.

Appendix A. Coefficient matrices

In this appendix, we present the flux vectors and the coefficient matrices of the hydrodynamic equations, as expressed in terms of the (physical) entropy variables.

For referential convenience, the mapping from U to V is provided here:

$$V = \frac{1}{T} \begin{Bmatrix} \mu - \frac{|u|^2}{2} \\ u_1 \\ u_2 \\ u_3 \\ -1 \end{Bmatrix} = \frac{v}{T} \begin{Bmatrix} -U_5 + \frac{T}{v} [(\gamma + 1)c_v - s] \\ U_2 \\ U_3 \\ U_4 \\ -U_1 \end{Bmatrix}, \quad (\text{A.1})$$

where

$$s = c_p \ln \frac{T}{T_0} - R \ln \frac{P}{P_0} + s_0, \quad (\text{A.2})$$

$$\frac{T}{v} = \frac{1}{c_v} \left[U_5 - \frac{U_2^2 + U_3^2 + U_4^2}{2U_1} \right]. \quad (\text{A.3})$$

The inverse mapping $V \rightarrow U$ is given as

$$U = \frac{1}{v} \begin{Bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e_{\text{int}} + \frac{|u|^2}{2} \end{Bmatrix} = \frac{T}{v} \begin{Bmatrix} -V_5 \\ V_2 \\ V_3 \\ V_4 \\ c_v - \frac{(V_2^2 + V_3^2 + V_4^2)}{2V_5} \end{Bmatrix}, \quad (\text{A.4})$$

where

$$\frac{T}{v} = K^{1/R} \frac{1}{(-V_5)^{c_p/R}} \exp\left(\frac{-s + s_0}{R}\right), \quad K = \frac{\left(\frac{P_0}{R}\right)^R}{(T_0)^{c_p}}, \quad (\text{A.5})$$

$$s = \gamma c_v - V_1 + \frac{V_2^2 + V_3^2 + V_4^2}{2V_5}. \quad (\text{A.6})$$

The coefficient matrices are expressed with the help of the following variables:

$$h = c_p T, \quad e_{\text{int}} = c_v T, \quad \alpha_p = \frac{1}{T}, \quad \beta_T = \frac{1}{P}, \quad c_p - c_v = R, \quad (\text{A.7})$$

$$k = \frac{|u|^2}{2}, \quad d = \frac{v\alpha_p T}{\beta_T}, \quad \bar{\gamma} = \frac{v\alpha_p}{\beta_T c_v}, \quad (\text{A.8})$$

$$c_1 = u_1^2 + \frac{v}{\beta_T}, \quad c_2 = u_2^2 + \frac{v}{\beta_T}, \quad c_3 = u_3^2 + \frac{v}{\beta_T}, \quad (\text{A.9})$$

$$\bar{c}_1 = u_1^2 + c_v T, \quad \bar{c}_2 = u_2^2 + c_v T, \quad \bar{c}_3 = u_3^2 + c_v T, \quad (\text{A.10})$$

$$e_1 = h + k, \quad e_2 = e_1 - d, \quad e_3 = e_2 + \frac{v}{\beta_T}, \quad e_4 = e_2 + 2 \frac{v}{\beta_T}, \quad (\text{A.11})$$

$$\bar{e}_1 = h - k, \quad \bar{e}_2 = \bar{e}_1 - d, \quad \bar{e}_3 = \bar{e}_2 - c_v T, \quad a^2 = \frac{v c_p}{c_v \beta_T}, \quad (\text{A.12})$$

$$u_{12} = u_1 u_2, \quad u_{23} = u_2 u_3, \quad u_{31} = u_3 u_1, \quad u_{123} = u_1 u_2 u_3, \quad (\text{A.13})$$

$$e_5 = e_1^2 - 2e_1 d + \frac{v(2k + c_p T)}{\beta_T}, \quad \bar{e}_5 = \bar{e}_1^2 - 2\bar{e}_1 d + 2k c_v T + \frac{v c_p T}{\beta_T}, \quad (\text{A.14})$$

$$\tilde{A}_0 = \frac{\beta_T T}{v^2} \begin{bmatrix} 1 & u_1 & u_2 & u_3 & e_2 \\ & c_1 & u_{12} & u_{31} & u_1 e_3 \\ & & c_2 & u_{23} & u_2 e_3 \\ \text{symm} & & & c_3 & u_3 e_3 \\ & & & & e_5 \end{bmatrix}, \quad (\text{A.15})$$

$$\tilde{A}_0^{-1} = \frac{v}{c_v T^2} \begin{bmatrix} \bar{e}_5 & u_1 \bar{e}_3 & u_2 \bar{e}_3 & u_3 \bar{e}_3 & -\bar{e}_2 \\ & \bar{c}_1 & u_{12} & u_{31} & -u_1 \\ & & \bar{c}_2 & u_{23} & -u_2 \\ \text{symm} & & & \bar{c}_3 & -u_3 \\ & & & & 1 \end{bmatrix}. \quad (\text{A.16})$$

The advective Jacobians with respect to U , $A_i = F_{i,U}$, are given by

$$A_1 = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ a^2 - u_1^2 - \bar{e}_1 \bar{\gamma} & -u_1(\bar{\gamma} - 2) & -u_2 \bar{\gamma} & -u_3 \bar{\gamma} & \bar{\gamma} \\ -u_{12} & u_2 & u_1 & 0 & 0 \\ -u_{31} & u_3 & 0 & u_1 & 0 \\ -u_1(e_1 + \bar{e}_1 \bar{\gamma} - a^2) & e_1 - u_1^2 \bar{\gamma} & -u_{12} \bar{\gamma} & -u_{31} \bar{\gamma} & u_1(\bar{\gamma} + 1) \end{bmatrix}, \quad (\text{A.17})$$

$$A_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ u_{12} & u_2 & u_1 & 0 & 0 \\ a^2 - u_2^2 - \bar{e}_1 \bar{\gamma} & -u_1 \bar{\gamma} & -u_2(\bar{\gamma} - 2) & u_3 \bar{\gamma} & \bar{\gamma} \\ -u_{23} & 0 & u_3 & u_2 & 0 \\ -u_2(e_1 + \bar{e}_1 \bar{\gamma} - a^2) & -u_{12} \bar{\gamma} & e_1 - u_2^2 \bar{\gamma} & -u_{23} \bar{\gamma} & u_2(\bar{\gamma} + 1) \end{bmatrix}, \quad (\text{A.18})$$

$$A_3 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ -u_{31} & u_3 & 0 & u_1 & 0 \\ -u_{23} & 0 & u_3 & u_2 & 0 \\ a^2 - u_3^2 - \bar{e}_1 \bar{\gamma} & -u_1 \bar{\gamma} & -u_2 \bar{\gamma} & -u_3(\bar{\gamma} - 2) & \bar{\gamma} \\ -u_3(e_1 + \bar{e}_1 \bar{\gamma} - a^2) & -u_{31} \bar{\gamma} & -u_{23} \bar{\gamma} & e_1 - u_3^2 \bar{\gamma} & u_3(\bar{\gamma} + 1) \end{bmatrix}. \quad (A.19)$$

The advective Jacobian matrices with respect to V , $\tilde{A}_i = F_{i,V} = \tilde{A}_i \tilde{A}_0$, are given by

$$\tilde{A}_1 = \frac{\beta_T T}{v^2} \begin{bmatrix} u_1 & c_1 & u_{12} & u_{31} & u_1 e_3 \\ u_1 \left(u_1^2 + 3 \frac{v}{\beta_T} \right) & u_2 c_1 & u_3 c_1 & e_1 \frac{v}{\beta_T} + u_1^2 e_4 & \\ \text{symm} & u_1 c_2 & u_{123} & u_{12} e_4 & \\ & u_1 c_3 & u_{31} e_4 & & \\ & & u_1 \left(e_5 + 2e_1 \frac{v}{\beta_T} \right) & & \end{bmatrix}, \quad (A.20)$$

$$\tilde{A}_2 = \frac{\beta_T T}{v^2} \begin{bmatrix} u_2 & u_{12} & c_2 & u_{23} & u_2 e_3 \\ & u_2 c_1 & u_1 c_2 & u_{123} & u_{12} e_4 \\ & & u_2 \left(u_2^2 + 3 \frac{v}{\beta_T} \right) & u_3 c_2 & e_1 \frac{v}{\beta_T} + u_2^2 e_4 \\ \text{symm} & & & u_2 c_3 & u_{23} e_4 \\ & & & & u_2 \left(e_5 + 2e_1 \frac{v}{\beta_T} \right) \end{bmatrix}, \quad (A.21)$$

$$\tilde{A}_3 = \frac{\beta_T T}{v^2} \begin{bmatrix} u_3 & u_{31} & u_{23} & c_3 & u_3 e_3 \\ & u_3 c_1 & u_{123} & u_1 c_3 & u_{31} e_4 \\ & & u_3 c_2 & u_2 c_3 & u_{23} e_4 \\ \text{symm} & & & u_3 \left(u_3^2 + 3 \frac{v}{\beta_T} \right) & e_1 \frac{v}{\beta_T} + u_3^2 e_4 \\ & & & & u_3 \left(e_5 + 2e_1 \frac{v}{\beta_T} \right) \end{bmatrix}. \quad (A.22)$$

The right-hand side coefficient matrices \tilde{K}_{ij} , where $\tilde{K}_{ij} V_{,j} = F_i^h$ are given by

$$\hat{K}_{ij} = K \delta_{ij}, \quad (A.23)$$

where

$$K = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ & 0 & 0 & 0 & 0 \\ & & 0 & 0 & 0 \\ \text{symm} & & & 0 & 0 \\ & & & & \frac{\kappa n}{m} T^2 \end{bmatrix}. \quad (A.24)$$

The source vector in terms of the V variables is given as

$$F = \frac{T}{v} \left\{ \begin{array}{c} 0 \\ \frac{\epsilon}{m} E_1 V_5 + \frac{\epsilon}{m \mu_{n0} T_0} \frac{V_2}{V_5} \\ \frac{\epsilon}{m} E_2 V_5 + \frac{\epsilon}{m \mu_{n0} T_0} \frac{V_3}{V_5} \\ \frac{\epsilon}{m} E_3 V_5 + \frac{\epsilon}{m \mu_{n0} T_0} \frac{V_4}{V_5} \\ -\frac{\epsilon}{m} (E_1 V_2 + E_2 V_3 + E_3 V_4) - \frac{\left(c_v - \frac{V_2^2 + V_3^2 + V_4^2}{2V_5} + \frac{3}{2} k_b \frac{T_0}{m} V_5 \right)}{\left[\frac{3}{2} \frac{\mu_{n0}}{\epsilon v_s^2} \frac{k_b T_0}{1 - V_5 T_0} - \frac{m \mu_{n0} T_0 V_5}{2\epsilon} \right]} \end{array} \right\}, \quad (\text{A.25})$$

where T/v is expressed in terms of V variables as given in (A.5). The source coefficient-matrix, \tilde{C} (where $\tilde{C}V = -F$), is not uniquely defined. One possible definition, which leads to a symmetric matrix, is

$$\tilde{C} = -\frac{T}{v} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\epsilon}{m \mu_{n0} T_0 V_5} & 0 & 0 & \frac{\epsilon E_1}{m} \\ 0 & 0 & \frac{\epsilon}{m \mu_{n0} T_0 V_5} & 0 & \frac{\epsilon E_2}{m} \\ 0 & 0 & 0 & \frac{\epsilon}{m \mu_{n0} T_0 V_5} & \frac{\epsilon E_3}{m} \\ 0 & \frac{\epsilon E_1}{m} & \frac{\epsilon E_2}{m} & \frac{\epsilon E_3}{m} & \bar{\omega} \end{bmatrix}, \quad (\text{A.26})$$

where

$$\bar{\omega} = -2 \frac{\epsilon}{m} \frac{(E_1 V_2 + E_2 V_3 + E_3 V_4)}{V_5} - \frac{\left(c_v - \frac{V_2^2 + V_3^2 + V_4^2}{2V_5} + \frac{3}{2} k_b \frac{T_0}{m} V_5 \right)}{\left[\frac{3}{2} \frac{\mu_{n0}}{\epsilon v_s^2} \frac{k_b T_0}{1 - V_5 T_0} - \frac{m \mu_{n0} T_0 V_5}{2\epsilon} \right] V_5}. \quad (\text{A.27})$$

Acknowledgment

This research was sponsored by DARPA through contract no. DAA100391-C-0043. The authors would like to thank Drs. Datong Chen, Ke Chih Wu, Zhiping Yu and Zdenek Johan for helpful discussions, and Dr. Farzin Shakib for providing us with the initial code for Euler and Navier–Stokes analysis. The equipment support provided by Digital Equipment Corporation to Professor K.H. Law is appreciated.

Notation

ε	electronic charge
κ	conductivity per unit volume
μ	specific chemical potential
μ_{n0}	low field electronic mobility
θ	dielectric permittivity
ψ	electrostatic potential
ψ_b	built-in potential
ψ_{appl}	applied bias
ψ^h	finite element approximation of potential
$\bar{\psi}^h$	finite element weighting function for Poisson problem
τ_p	momentum relaxation time
τ_w	energy relaxation time
Ω	spatial domain
Ω_n^e	element spatial domain at the n th time slab
γ	ratio of specific heats
v^h	discontinuity capturing factor
v	specific volume
τ	least squares matrix
Γ	boundary of spatial domain
Γ_g	boundary on which essential boundary conditions are prescribed
Γ_h	boundary on which natural boundary conditions are prescribed
A_i	Euler Jacobian matrix with respect to conservative variables in direction i
\tilde{A}_i	Euler Jacobian matrix with respect to entropy variables in direction i
\tilde{A}_0	Riemannian metric tensor
\tilde{B}_n	boundary of n th space time slab
\tilde{C}	source coefficient matrix
c_v	specific heat at constant volume
c_p	specific heat at constant pressure
E	electric field vector
E_i	electric field in direction i
e_{tot}	electron total energy per unit mass
e_{int}	electron internal energy per unit mass
F	source vector
F_i	Euler flux vector in direction i
F_i^h	heat flux vector in direction i
g	prescribed boundary condition vector for HD equations
g_p	prescribed boundary condition for Poisson problem
\mathcal{H}	entropy function
h_i	prescribed natural boundary condition in direction i for Poisson problem
I_n	time interval
K	diffusivity matrix with respect to conservative variables
\tilde{K}	diffusivity matrix with respect to entropy variables

k_b	Boltzmann constant
L	reference length
$M^{(i)}$	consistent tangent matrix at i th iterative step for HD system
m	electron mass
N_D^+	concentration of ionized donor
N_A^-	concentration of ionized acceptor
$N_A^{(n)}$	finite element spatial shape function of node A for the n th time slab
n	concentration of electrons
n_d	doping concentration
n_i	intrinsic concentration of electrons
n_{np}	number of nodal points
n_{dof}	number of degrees of freedom for HD system
p_e, p	electron momentum density vector
p_h	hole momentum density vector
p	concentration of holes
P	electron pressure per unit mass
t	time
Q_n	space time slab at time level n
Q_n^e	element space time slab at time level n
q_e, q	electron heat flux vector
q_h	hole heat flux vector
q_i	heat flux in direction i
R^i	residual vector at i th iterative step for HD system
R	specific gas constant
s	thermodynamic entropy
T_e, T	temperature of electrons
T_h	temperature of holes
T_0	temperature of the lattice
U	conservative variables vector
u_e, u	electron velocity vector
u_h	hole velocity vector
u_i	velocity in direction i
v_s	saturation velocity
V	entropy variable vector
V^h	finite element trial solution vector
V_t	reference voltage
v_A	vector of nodal unknowns at node A for HD system
W^h	weighting function vector for hydrodynamic equations
w_e, w	electron energy density
w_h	hole energy density
w_A	weighting function vector at node A for HD system
$[]_{col}$	collision terms
$()^*$	nondimensional quantity
$()_0$	reference value

References

- [1] M.R. Pinto, Comprehensive semiconductor device simulation for silicon ULSI, Dept. of Elec. Engrg., Ph.D. Thesis, Stanford University, 1990.
- [2] D. Chen et al., Elimination of spurious velocity overshoot using a new energy transport model, unpublished.
- [3] G. Baccarani and M.R. Wordeman, An investigation of steady-state velocity overshoot in silicon, *Solid-State Electron.* 28 (1985) 407–416.
- [4] M. Rudan and F. Odeh, Multi-dimensional discretization scheme for the hydrodynamic model of semiconductor devices, *COMPEL* 5 (1986) 149–183.
- [5] M. Rudan, F. Odeh and J. White, Numerical solution of the hydrodynamic model for a one-dimensional device, *COMPEL* 6 (1987) 151–170.
- [6] C.L. Gardner, J.W. Jerome and D.J. Rose, Numerical methods for the hydrodynamic device model: Subsonic flow, *IEEE Trans. Comput. Aided Design* 8 (1989) 501–507.
- [7] E. Fatemi, J.W. Jerome and S. Osher, Solution of the hydrodynamic device model using high-order nonoscillatory shock capturing algorithms, *IEEE Trans. Comput. Aided Design* 10 (1991) 232–244.
- [8] T.J.R. Hughes, Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier–Stokes equations, *Internat. J. Numer. Methods Engrg.* 7 (1987) 1261–1275.
- [9] T.J.R. Hughes, M. Mallet and A. Mizukami, A new finite element formulation for computational fluid dynamics: II. Beyond SUPG, *Comput. Methods Appl. Mech. Engrg.* 54 (1986) 341–355.
- [10] F. Shakib, Finite element analysis of the compressible Euler and Navier–Stokes equations, Dept. of Mech. Engrg., Ph.D. Thesis, Stanford University, 1988.
- [11] Z. Johan, Data parallel finite element techniques for large-scale computational fluid dynamics, Dept. of Mech. Engrg., Ph.D. Thesis, Stanford University, 1992.
- [12] K. Blotekjaer, Transport equations for electrons in two-valley semiconductors, *IEEE Trans. Electron. Devices* 17 (1970) 38–47.
- [13] S. Selberherr, *An Analysis and Simulation of Semiconductor Devices* (Springer, New York, 1984).
- [14] A. Harten, On the symmetric form of systems of conservation laws with entropy, *J. Comput. Phys.* 49 (1983) 151–164.
- [15] T.J.R. Hughes, L.P. Franca and M. Mallet, A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible Euler and Navier–Stokes equations and the second law of thermodynamics, *Comput. Methods Appl. Mech. Engrg.* 54 (1986) 223–234.
- [16] F. Chalot, T.J.R. Hughes and F. Shakib, Symmetrization of conservation laws with entropy for high-temperature hypersonic computations, *Comput. Systems Engrg.* 1 (1990) 494–521.
- [17] A. Mizukami and T.J.R. Hughes, A Petrov–Galerkin finite element method for convection-dominated flows: an accurate upwinding technique for satisfying the maximum principle, *Comput. Methods Appl. Mech. Engrg.* 50 (1985) 181–193.
- [18] C. Johnson, Streamline diffusion methods for problems in fluids, in: R.H. Gallagher et al., eds, *Finite Elements in Fluids*, Vol. VI (Wiley, London, 1986) 251–261.
- [19] T.J.R. Hughes and M. Mallet, A new finite element formulation for computational fluid dynamics: IV. A discontinuity-capturing operator for multidimensional advective-diffusive systems, *Comput. Methods Appl. Mech. Engrg.* 58 (1986) 329–336.
- [20] A.C. Galeão and E.G. Dutra Do Carmo, A consistent approximate upwind Petrov–Galerkin method for convection-dominated problems, *Comput. Methods Appl. Mech. Engrg.* 68 (1988) 83–95.
- [21] T.J.R. Hughes, L.P. Franca and G.M. Hulbert, A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive equations, *Comput. Methods Appl. Mech. Engrg.* 73 (1989) 173–189.
- [22] T.J.R. Hughes and M. Mallet, A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems, *Comput. Methods Appl. Mech. Engrg.* 58 (1986) 305–328.
- [23] T.J.R. Hughes, *The Finite Element Method: Linear Static and Dynamic Finite Element Analysis* (Prentice Hall, Englewood Cliffs, NJ, 1987).
- [24] R.L. Lee, P.M. Gresho and R.L. Sani, Smoothing techniques for certain primitive variable solutions of the Navier–Stokes equations, *Internat. J. Numer. Methods Engrg.* 14 (1979) 1785–1804.
- [25] C. Johnson, U. Nävert and J. Pitkäranta, Finite element methods for linear hyperbolic problems, *Comput. Methods Appl. Mech. Engrg.* 45 (1984) 285–312.

IEICE **TRANSACTIONS**

on Electronics

VOL. E77-C
NO. 2
FEBRUARY 1994



The Institute of Electronics, Information and Communication Engineers
Kikai-Shinko-Kaikan Bldg., 5-8, Shibakoen 3chome, Minato-ku, Tokyo, 105 Japan

PAPER *Special Issue on 1993 VLSI Process and Device Modeling Workshop (VPAD93)*

Space-Time Galerkin/Least-Squares Finite Element Formulation for the Hydrodynamic Device Equations

N. R. ALURU[†], Kincho H. LAW[†], Peter M. PINSKY[†], Arthur RAEFSKY[†],
Ronald J. G. GOOSSENS[†] and Robert W. DUTTON[†], *Nonmembers*

SUMMARY Numerical simulation of the hydrodynamic semiconductor device equations requires powerful numerical schemes. A Space-time Galerkin/Least-Squares finite element formulation, that has been successfully applied to problems of fluid dynamics, is proposed for the solution of the hydrodynamic device equations. Similarity between the equations of fluid dynamics and semiconductor devices is discussed. The robustness and accuracy of the numerical scheme are demonstrated with the example of a single electron carrier submicron silicon MESFET device.

key words: semiconductor devices, hydrodynamic model, Galerkin/least-squares finite element method, space-time formulation

1. Introduction

Integrated-circuit technology is increasingly complex and costly. Traditional empirical approaches to determine the electrical characteristics of semiconductor devices are no longer viable. An alternative approach is to employ numerical simulations. A series of numerical simulations for different operating conditions are typically required before the final I - V curve characterizing the device behavior can be obtained. Hence, reliable and accurate numerical simulations are of utmost importance to device modeling. Having realized this, there has been a significant amount of effort spent in developing robust and accurate numerical device simulators. Most simulation programs employed today are based on finite difference or finite volume approaches. These methods become quite complex for problems with unstructured grids. It has been long recognized that the finite element method is a powerful tool for solving a system of partial differential equations. In this paper, we propose a finite element scheme that has been proven very effective in the area of computational fluid dynamics and demonstrate its applicability and advantages in semiconductor device simulations.

Device simulation tools have been based primarily on the drift-diffusion (DD) model for carrier transport, a simplification of the Boltzmann Transport Equation (BTE). With the scaling of silicon devices

into deep submicron region, non-stationary phenomena such as velocity overshoot and carrier heating are becoming increasingly important to determine the characteristic of these devices. Due to the assumption of local equilibrium, the DD model cannot capture such non-stationary phenomena accurately. Although the direct solution of BTE, for example via Monte Carlo method, can capture the above phenomena, the noise in the solution and the computational cost prevent it from wide usage for device simulation. An attractive alternative is to employ full Hydrodynamic (HD) [1]-[3] or HD-like models. HD-like models are obtained by adding an energy equation to the DD model. HD-like model has little resemblance to fluid equations, and is more appropriately referred as the Energy Transport (ET) model [4]. The full HD model can be directly derived from the zero, first and second moments of the BTE with a few simplifying assumptions [1]. These equations have a direct analogy to fluid dynamics equations. As shown in this paper the HD equations for device simulation resemble the equations of compressible gas flow. The development of a robust and accurate numerical scheme for the full HD model is the subject of this paper.

The goal of this paper is to present a stable and robust finite element method for the HD equations based on a space-time Galerkin/Least-Squares formulation. This paper is organized as follows: Section 2 summarizes the partial differential equations for semiconductor device simulation and states the assumptions used in the derivation. Section 3 gives a comparison of HD equations to the equations of fluid dynamics. Section 4 gives an overview of the finite element methods commonly used and presents the discretization procedure employed in this work. Section 5 discusses the solution strategy employed. Section 6 presents numerical results and Sect. 7 summarizes the paper.

2. Field Equations

Semiconductor devices can be simulated by solving the coupled Poisson and HD equations. For single carrier devices, the transport equations for electron gas described by the HD model are summarized as follows:

Manuscript received July 29, 1993.

Manuscript revised October 8, 1993.

[†] The authors are with the Applied Electronics Laboratory, Stanford University, Stanford, California 94305, U.S.A.

$$\frac{\partial n}{\partial t} + \nabla \cdot (n\mathbf{u}) = \left[\frac{\partial n}{\partial t} \right]_{col} \quad (1)$$

$$\frac{\partial \mathbf{p}}{\partial t} + \mathbf{u}(\nabla \cdot \mathbf{p}) + (\mathbf{p} \cdot \nabla) \mathbf{u} = -\epsilon n \mathbf{E} - \nabla (nk_b T) + \left[\frac{\partial \mathbf{p}}{\partial t} \right]_{col} \quad (2)$$

$$\frac{\partial w}{\partial t} + \nabla \cdot (\mathbf{u}w) = -\epsilon n (\mathbf{u} \cdot \mathbf{E}) - \nabla \cdot (\mathbf{u}nk_b T) - \nabla \cdot \mathbf{q} + \left[\frac{\partial w}{\partial t} \right]_{col} \quad (3)$$

Equations (1), (2), and (3) are the particle continuity and conservation laws for electron momentum and energy, respectively. In the above equations, n is the concentration of electrons; \mathbf{u} is the electron velocity vector; \mathbf{p} is the electron momentum density vector; T is the electron temperature; w is the electron energy density; \mathbf{q} is the electron heat flux vector; ϵ is the magnitude of an elementary charge; k_b is the Boltzmann constant and $[]_{col}$ denotes collision terms. Equations (1)-(3) represent a system of three partial differential equations with 5 unknowns- n , \mathbf{u} , \mathbf{p} , T and w . The following assumptions can be made regarding the equations [1], [5]:

- i) The carrier temperature, T is assumed to be a scalar quantity.
- ii) The collision terms are approximated by relaxation times.
- iii) The energy bands are assumed to be parabolic i.e. the effective mass of the carrier (electron in our discussion) is a scalar constant, m . With this assumption, the following constitutive relations can be given for the momentum and energy density.

$$\mathbf{p} = m n \mathbf{u} \quad (4)$$

$$w = \frac{3}{2} n k_b T + \frac{1}{2} m n |\mathbf{u}|^2 \quad (5)$$

- iv) The heat conduction is assumed to be given by the Fourier law i.e.

$$\mathbf{q} = -\chi \nabla T \quad (6)$$

- v) The heat conductivity is assumed to be given by the Wiedemann-Franz law i.e.

$$\chi = \left(\frac{5}{2} + \vartheta \right) \frac{\mu_{n0} n k_b^2 T_0}{\epsilon} \quad (7)$$

where μ_{n0} is the low field electron mobility, T_0 is the temperature of the lattice and ϑ is a parameter associated with the energy dependence of the momentum relaxation time. A value of $\vartheta = -1$ is employed in this model, which assumes that the mobility is inversely proportional to the carrier temperature.

With assumption (iii), the five unknowns are reduced to three i.e. n , \mathbf{u} and T , and the system can be solved

given the expressions for the collision terms.

The collision terms $[]_{col}$ in Eqs. (1), (2), and (3) describe the rate of change of concentration, momentum and energy due to collisions. For single-carrier devices, there are no generation and recombination processes. Thus, the collision term for the rate of change of particle/concentration vanishes:

$$\left[\frac{\partial n}{\partial t} \right]_{col} = 0 \quad (8)$$

The collision terms in the momentum conservation, Eq. (2), and the energy conservation, Eq. (3), represent the rate of change of momentum and energy density, respectively, due to intraband collisions. These can be expressed in terms of momentum and energy relaxation times as [2]

$$\left[\frac{\partial \mathbf{p}}{\partial t} \right]_{col} = \frac{-\mathbf{p}}{\tau_p} \quad (9)$$

$$\left[\frac{\partial w}{\partial t} \right]_{col} = \frac{-\left(w - \frac{3}{2} n k_b T_0 \right)}{\tau_w}$$

where the momentum relaxation time is expressed as

$$\tau_p = m \frac{\mu_{n0}}{\epsilon} \frac{T_0}{T} \quad (10)$$

and the energy relaxation time is expressed as

$$\tau_w = \frac{3}{2} \frac{\mu_{n0}}{\epsilon v_s^2} \left(\frac{k_b T T_0}{T + T_0} \right) + \frac{\tau_p}{2} \quad (11)$$

and v_s is the saturation velocity.

The electron concentration is coupled to the electrostatic potential by the Poisson equation. The Poisson equation, derived from Maxwell's equations [6], is given by

$$\nabla \cdot (\theta \mathbf{E}) = -\epsilon (n - N_D^+) \quad (12)$$

where the electric field \mathbf{E} is related to the electrostatic potential ψ by,

$$\mathbf{E} = -\nabla \psi \quad (13)$$

In the above equations, N_D^+ is the concentration of ionized donor and θ is the dielectric permittivity. In deriving the Poisson equation (12) from the Maxwell's equations, a time independent and isotropic dielectric permittivity is assumed and the magnetic field effects are neglected.

3. HD Equations vs. Equations of Fluid Dynamics

In a macroscopic approach to fluid dynamics, there are two well known models: the Euler or ideal model (in which the fluid pressure is given by the isotropic part of the stress tensor and the heat conduction is assumed negligible) and the Navier-Stokes model (in which the fluid pressure is a tensor comprising of viscous terms

and the heat conduction is given by the Fourier law). The macroscopic equations for both the models can be derived from the Boltzmann equation with suitable assumptions. Both Euler and Navier-Stokes equations can be physically interpreted as the conservation of mass, momentum and energy.

In the previous section, we have noted that the HD device model has also physically resulted from the conservation of particle, momentum and energy. A natural question that arises at this juncture is, whether there is any similarity or relationship between these two systems. A close examination reveals that the two systems are similar, but the HD equations are not identical to either the Euler equations or the Navier-Stokes equations. While the HD equations do not contain viscous terms in the model that we consider, they are not the same as the Euler equations because of the presence of heat conduction term in the energy equation. Furthermore, the highly non-linear source terms in the HD model are absent in fluid modeling with the Euler and the Navier-Stokes equations. As shown below, the existing similarity between the two sets of conservation laws can be derived by introducing two new quantities: electron pressure and electron total energy.

Let's define the electron pressure per unit mass, P , and energy density, w , as follows:

$$P = \frac{nk_b T}{m} \quad (14)$$

$$w = nme_{tot} \quad (15)$$

where e_{tot} is the total energy per unit mass. Substituting these terms in Eq. (1)-(3), we obtain the modified system of equations given in Eq. (16).

$$\begin{bmatrix} n \\ nu_1 \\ nu_2 \\ nu_3 \\ ne_{tot} \end{bmatrix}_{,i} + \begin{bmatrix} nu_i \\ nu_i u_1 + P\delta_{1i} \\ nu_i u_2 + P\delta_{2i} \\ nu_i u_3 + P\delta_{3i} \\ nu_i e_{tot} + Pu_i \end{bmatrix}_{,i} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_i \end{bmatrix}_{,i} + \begin{bmatrix} 0 \\ n\left(-\frac{\epsilon}{m}E_1 - \frac{u_1}{\tau_p}\right) \\ n\left(-\frac{\epsilon}{m}E_2 - \frac{u_2}{\tau_p}\right) \\ n\left(-\frac{\epsilon}{m}E_3 - \frac{u_3}{\tau_p}\right) \\ -\frac{\epsilon nu_i E_i}{m} - \frac{\left(ne_{tot} - \frac{3}{2m}nk_b T_0\right)}{\tau_w} \end{bmatrix}_{,i} \quad (16)$$

This system represents a perfect gas flow with γ , the ratio of specific heats equal to 5/3. In summary, the

electron HD equations resemble the flow of a real compressible fluid given by Euler equations, in the presence of electric field and with the addition of a heat conduction term and the highly nonlinear source terms.

The hydrodynamic equations discussed above can be put in the form of a general system. The idea is to identify the role played by each of the terms in the equations and categorize them as convective, diffusive or source contributions. In doing so, efficient numerical schemes can be designed by a good understanding of the contribution of each of the terms in the equations. Furthermore, the numerical scheme can be generalized to solve any set of partial differential equations that can be cast in a similar form. Specifically, the HD equations (16) can be written in a system form as

$$U_{,t} + F^c_{i,i} = F^h_{i,i} + F \quad (17)$$

where U is the vector of physical variables, $F^c(U)$ is a vector containing convective terms, $F^h(U)$ is the vector containing diffusive terms, and $F(U)$ contains the source or driving terms. The above equation is commonly referred to as the Advective-Diffusive (AD) or Convective-Diffusive (CV) system of equations. It is interesting to note that partial differential equations describing the physics of fluid dynamics (Euler/Navier-Stokes equations), shallow water equations, semiconductor device equations and others can be written in the same form of (16), with slight variants in the definition of the vectors. For the present case, the explicit definitions of the vectors are given in [7]. Equation (16) is the starting point for several numerical schemes. One may wish to rewrite (16) in a different form according to the numerical scheme employed.

4. Finite Element Methods

The numerical and mathematical treatment of semiconductor device equations employing finite element methods has long been considered an enigma. Even for the simplest DD model, it was recognized that the standard Galerkin finite element method does not work well. This is not surprising, as the stencil resulting from the Galerkin finite element method is very similar to the one resulting from the central difference methods (in fact they are identical in the absence of sources for linear basis functions). The problems faced in the central difference method can be corrected by employing upwind schemes [8]. Similar techniques have been developed for finite element methods for fluids. In the following, a brief overview is presented on the problems faced in the Galerkin finite element method and the evolution of several different new schemes to correct the problems faced in the Galerkin method.

Standard Galerkin finite element method exhibits spurious oscillations for advective-diffusive type equations when the physical diffusion present in the system is very small. While the method works well for "large" diffusion, the "negligible" amount of diffusion present poses serious problems to the numerical schemes, since it causes sharp layers in the solution. When these sharp layers are not captured properly, numerical results obtained are "polluted" and are inaccurate. This is a well known effect when the Galerkin finite element method is applied for small diffusion problems or hyperbolic systems.

To overcome the problems of Galerkin method, the "classical artificial diffusion and upwind method" which is analogous to the artificial diffusion and upwind difference method was proposed [9]. In this scheme, artificial diffusion is added to the already existing physical diffusion, to provide the necessary stability. However, this method is not consistent as it is not a weighted residual formulation and it is only first-order accurate giving overly diffusive results. While this method suggests that the artificial diffusion approach is needed to provide stability, the question that remained was the optimum amount of artificial diffusion that should be added to retain a weighted residual formulation and to attain higher order accuracy with sufficient stability.

In a major advancement [10], the Streamline-upwind/Petrov Galerkin (SUPG) method is developed to rectify many of the problems faced earlier in Galerkin and classical artificial diffusion methods. This method can be viewed as a simple extension to the Galerkin method. The essential idea in this method is to add artificial diffusion only in the flow direction, thus providing higher order accuracy. Unlike the classical artificial diffusion method, SUPG is based on a weighted residual formulation and hence it is a consistent method. Stated differently, in SUPG finite element method, different trial and test functions are employed. SUPG is a rich finite element method as it encompasses the properties of stability, consistency and accuracy and has sound mathematical properties [11]. However, SUPG does not prevent overshoot or undershoot phenomena occurring in the vicinity of sharp layers. These undershoot and overshoot phenomena can be controlled by introducing an additional 'discontinuity-capturing' term which acts in the direction of the solution gradient rather than the streamline.

SUPG was developed to increase the control over the advective derivative term. The method has been generalized to provide control over all the terms in the governing differential equation [12], [13]. This method is popularly referred to as Galerkin/Least-Squares (GLS) finite element method. While many of the properties of this method are analogous to SUPG method, it is conceptually simpler than the SUPG method. In fact, in the absence of sources and when

linear basis functions are employed, both these methods are identical. In this method, terms of a least-squares type are added to the variational form obtained from the Galerkin method. These least-squares terms vanish when the exact solution is obtained, thus making it a consistent method. GLS is a higher order accurate method with good stability properties.

GLS is currently used for a wide variety of partial differential equations encountered in fluid and solid mechanics. Motivated by the success of this method and the resemblance of HD equations to Navier-Stokes equations, we enhance this method to account for the strong nonlinear source terms and apply it to the HD equations for semiconductor devices. The temporal behavior of HD equations is discretized using a discontinuous Galerkin method in time [14]. With a discontinuous Galerkin in time and Galerkin/Least-Squares in space this discretization scheme is known as space-time Galerkin/Least squares finite element formulation. The basic formulation of the space-time GLS discretization scheme can be summarized in the following steps:

- i) First, we state the weak form of the given partial differential equation (the strong form) by multiplying the strong form with an arbitrary test function. We then integrate the resulting system by parts. It can be shown that the strong form and the weak form are equivalent and the solution to the weak form is also the solution to the strong form (i.e. the governing partial differential equations).
- ii) To enhance the numerical stability, we introduce a least-squares term of a residual type to the weak form. Furthermore, a discontinuity-capturing term is added to overcome the undershoot and overshoot problems. The least-squares and discontinuity-capturing terms vanish when the exact solution is substituted in the weak form.
- iii) We employ the trial and test functions to be a combination of linear basis functions and substitute them into the nonlinear FEM equations.
- iv) The nonlinear system is solved using a Newton iterative scheme by linearizing the nonlinear equations with respect to the unknown trial solution.

A comprehensive mathematical treatment on the development of the finite element space-time Galerkin/Least-Squares formulation for the HD semiconductor device equations is given in Ref.[7]. For the Poisson problem, which is elliptic in nature, a standard Galerkin finite element method is employed.

5. Solution Scheme

A staggered scheme depicted as shown in Fig. 1 is applied to solve the coupled Poisson and HD equations. This scheme resembles the popular Gummel procedure referred to in the literature [15]. The Pois-

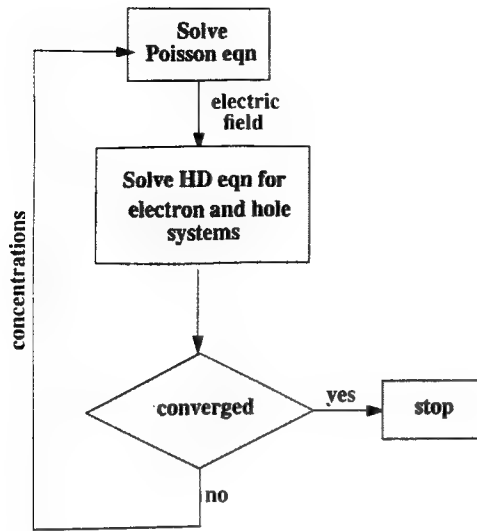


Fig. 1 A Staggered solution strategy for solving coupled hydrodynamic semiconductor device equations.

son equation and HD equations are solved in an uncoupled manner. The Poisson equation is first solved for the electrostatic potential and the electric field and the computed electric field values are used in the HD equations to solve for concentration, velocities and temperature. The concentration obtained from the HD equations provides a new source term to the Poisson equation. This procedure of alternatively solving the Poisson and HD equations is repeated until both the equations are solved to a desired tolerance.

6. Numerical Results

HD model and the space-time GLS numerical scheme discussed in this paper are tested on a submicron silicon Metal-Semiconductor Field-Effect Transistor (MESFET) at room temperature. The MESFET device, shown in Fig. 2, consists of a barrier junction at the input that acts as a control electrode (or gate), and two ohmic contacts, described as source and drain electrodes, through which the output current flows. The device is a special form of a junction field-effect transistor (JFET).

The three terminal device is $0.6 \mu\text{m}$ long along the x -direction and $0.2 \mu\text{m}$ wide along the y -direction. The contacts are placed on the top portion of the geometry. The source and drain contacts are approximately $0.1 \mu\text{m}$ long and the gate contact is approximately $0.2 \mu\text{m}$ long. The source and the drain contacts are separated from the gate contact by approximately $0.1 \mu\text{m}$. The substrate of the device is doped n -type with a doping value of $1.0 \times 10^{17}/\text{cm}^3$. The two n^+ regions shown in Fig. 2 are approximately of size $0.1 \mu\text{m} \times 0.05 \mu\text{m}$. The doping value in these regions is $3.0 \times 10^{17}/\text{cm}^3$ with abrupt junctions between n^+ and n boundaries. This example is similar to the device presented in [16],

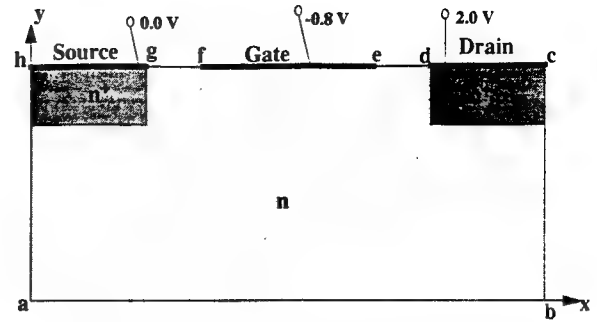


Fig. 2 A two-dimensional MESFET device.

but the boundary conditions and the numerical scheme employed are different.

A uniform mesh consisting of 3072 nodes and 2945 elements is used with 95 elements placed along the x -direction and 31 elements placed along the y -direction. The boundary conditions used for this experiment are summarized as follows:

- for source (h-g) and drain (d-c) contacts, $n = 3.0 \times 10^{17}/\text{cm}^3$, $u = 0 \text{ cm/s}$, $T = 300 \text{ K}$, and $\psi = \psi_b + \psi_{appl}$
- for gate contact f-e, $n = n_g$, $u = 0 \text{ cm/s}$ and $T = 300 \text{ K}$, and $\psi = \psi_{geff} = \psi_b - \psi_{appl}$
- on all other boundaries, $J_n = nu_n = 0$

The variable n_g denotes the concentration prescribed on the gate contact and is computed using the following expression

$$n_g = n_i e^{e\psi_{geff}/k_B T} \quad (18)$$

where n_i is the intrinsic concentration and ψ_{geff} denotes the effective potential applied on the gate. ψ_b denotes the built-in potential, ψ_{appl} denotes the potential applied on the source and drain contacts, and ψ_{geff} denotes the potential applied on the gate. The built-in potential is computed using the expression $\psi_b = (k_B T/e) \ln(n_d/n_i)$, where n_d denotes the doping.

The initial conditions used for this problem are $n = n_d \text{ cm}^{-3}$, $u = v = 0 \text{ cm/s}$, and $T = 300 \text{ K}$. Numerical experiments are performed for the following applied voltages: no potential is applied on the source, 2.0 V is applied on the drain, and -0.8 V is applied on the gate. The results for this example are presented in Figs. 3 to 9. The typical CPU time to obtain the steady state solution takes about 3-4 hours per a bias increment of 0.1 V on an IBM RS6000 workstation.

The concentration profile shown in Fig. 3 indicates two rapidly varying concentration regions. The first is between the source and gate and the second is between the gate and drain. In both of these regions the concentration varies by approximately 17 orders of magnitude. By effectively capturing these shocks, we have demonstrated the robustness of our scheme. The horizontal and vertical velocity profiles are shown in Figs. 4 and 5, respectively. From these plots it is clear that there is negligible current near the gate. The temperature profile shown in Fig. 6 shows a peak

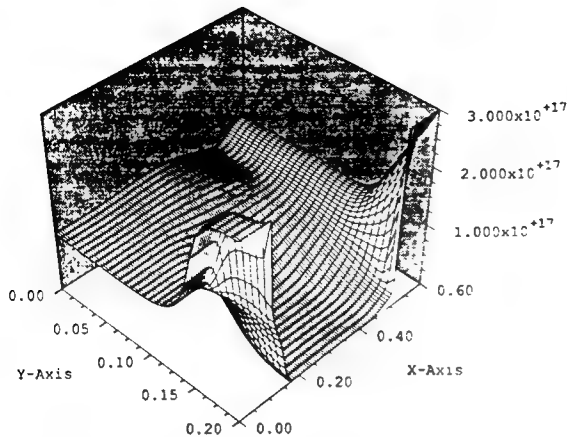
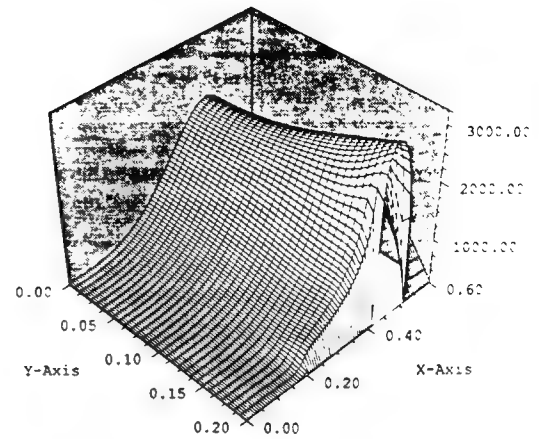
Fig. 3 Electron concentration (cm^{-3}).

Fig. 6 Electron temperature (K).

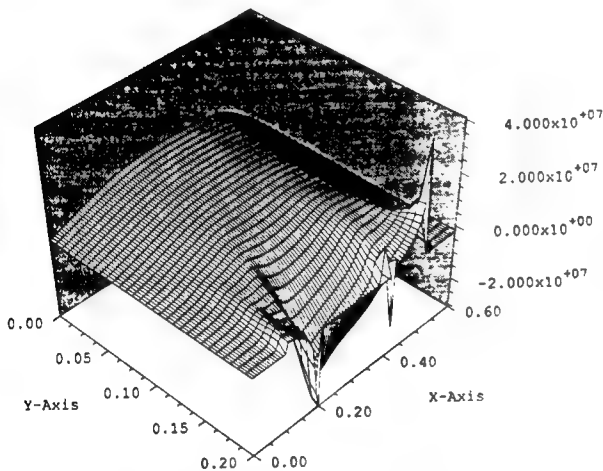


Fig. 4 Horizontal velocity of electrons (cm/s).

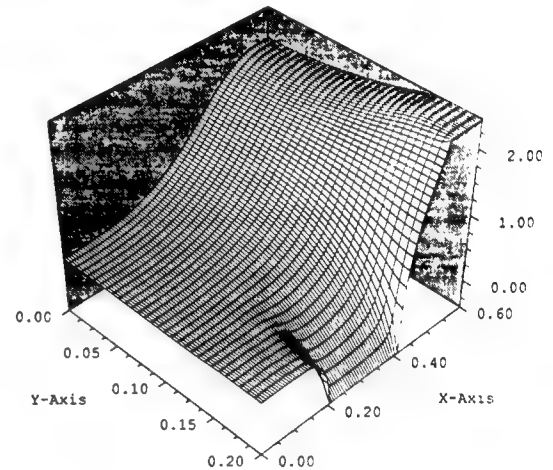


Fig. 7 Potential (V) distribution.

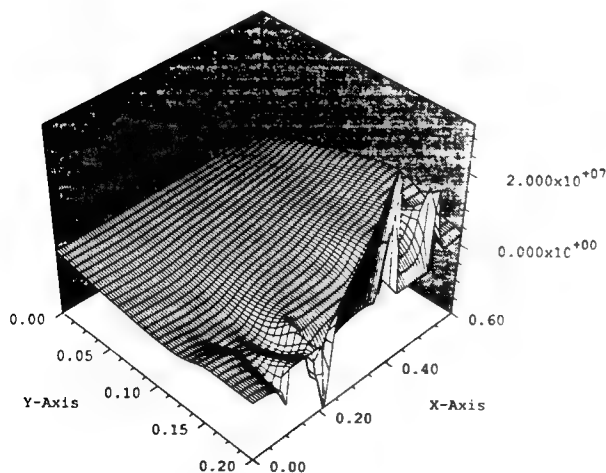


Fig. 5 Vertical velocity of electrons (cm/s).

towards the drain end of the device. The peak temperature reaches approximately 3250 K. Potential, shown in Fig. 7, varies rapidly at the vicinity very close to the contacts as well as in between the contacts. Smooth variation in the potential can be observed in the sub-

strate region. The horizontal and vertical profiles of electric fields are shown in Figs. 8 and 9, respectively. In these results, we have shown one set of boundary conditions. We have also simulated the same device with different boundary conditions and very different global solutions are observed.

7. Conclusions

A space-time Galerkin/Least-Squares finite element method is proposed for the solution of semiconductor hydrodynamic equations. The similarity between the HD equations and the Euler/Navier-Stokes equations of fluid dynamics is established. Results are shown for a single electron carrier two-dimensional silicon MES-FET device to demonstrate the robustness and accuracy of our numerical scheme.

This scheme extends in a straight forward manner to two carrier problems. Numerical simulation of two carrier devices will be addressed in a forth coming paper. Future efforts involve simulation of three-dimensional complex two carrier problems. Due to the complexity of the equations and the lack of complete

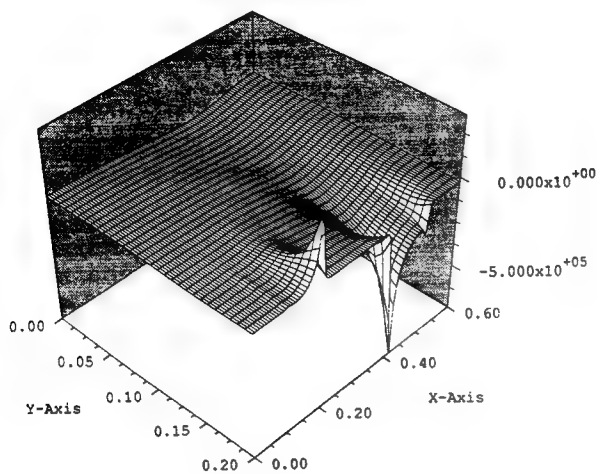


Fig. 8 Horizontal component of electric field (V/cm).

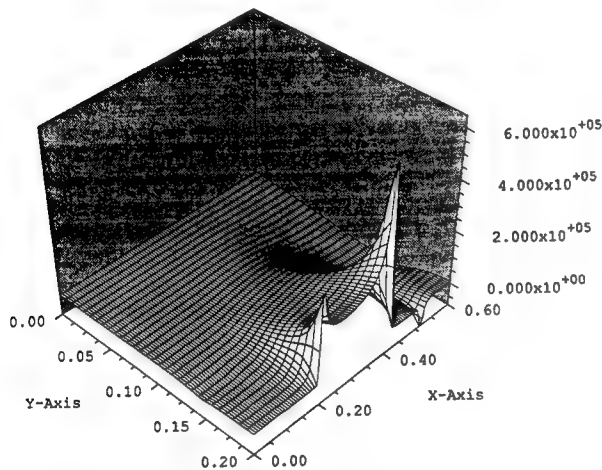


Fig. 9 Vertical component of electric field (V/cm).

understanding of well posed boundary conditions for full HD model, the convergence rate of the numerical scheme to reach a steady state solution could be slow for certain class of problems. To reduce the computer time for numerical simulation, our research effort will also include an implementation of the finite element program on high performance parallel computers.

Acknowledgments

This research was sponsored by ARPA through contract # DAAL 03-91-C-0043. The authors would like to thank Dr. Ke-Chih Wu for many helpful discussions and IBM corporation for providing our group with an RS6000 computer.

References

- [1] Blotekjaer, K., "Transport equations for electrons in two-valley semiconductors," *IEEE Trans. Electron Devices*, vol. ED-17, pp. 38-47, 1970.
- [2] Baccarani, G. and Wordeman, M. R., "An investigation of

- steady-state velocity overshoot in silicon," *Solid-State Electronics*, vol. 28, pp. 407-416, 1985.
- [3] Gardner, C. L., Jerome, J. W. and Rose, D. J., "Numerical methods for the hydrodynamic device model: subsonic flow," *IEEE Trans. Comput.-Aided Des. Integrated Circuits & Syst.*, vol. 8, pp. 501-507, 1989.
- [4] Chen, D., Kan, E. C., Ravaioli, U., Shu, C. and Dutton, R. W., "An improved energy transport model including non-parabolicity and non-maxwellian effects," *IEEE Electron Device Lett.*, vol. 13, no. 1, pp. 26-28, 1992.
- [5] Pinto, M. R., *Comprehensive semiconductor device simulation for silicon ULSI*, Dept. of Elec. Engg., Ph. D. Thesis, Stanford University, Aug. 1990.
- [6] Selberherr, S., *An analysis and simulation of semiconductor devices*, Springer-Verlag, New York, 1984.
- [7] Aluru, N. R., Raefsky, A., Pinsky, P. M., Law, K. H., Goossens, R. J. G. and Dutton, R. W., "A finite element formulation for the hydrodynamic semiconductor device equations," *Comp. Meth. Appl. Mech. Engg.*, vol. 107, pp. 269-298, 1993.
- [8] Scharfetter, D. L. and Gummel, H. K., "Large-signal analysis of a silicon read diode oscillator," *IEEE Trans. on Electron Devices*, vol. ED-16, pp. 64-77, 1969.
- [9] Gresho, P. M. and Lee, R. L., "Don't suppress the wiggles—They're telling you something," *Comp. and Fluids*, vol. 9, pp. 223-253, 1981.
- [10] Hughes, T. J. R. and Brooks, A., "A theoretical framework for Petrov-Galerkin methods with discontinuous weighting functions—Application to the streamline upwind procedure," in Gallagher et al., *FEM in Fluids*, vol. 4, pp. 47-65, 1982.
- [11] Hughes, T. J. R., Franca, L. P. and Mallet, M., "A new finite element formulation for computational fluid dynamics: VI. Convergence analysis of the generalized SUPG formulation for linear time-dependent multidimensional advective-diffusive systems," *Comp. Meth. Appl. Mech. Engg.*, vol. 63, pp. 97-112, 1987.
- [12] Hughes, T. J. R., Franca, L. P. and Hulbert, G. M., "A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive equations," *Comp. Meth. Appl. Mech. Engg.*, vol. 73, pp. 173-189, 1989.
- [13] Shakib, F., *Finite element analysis of the compressible Euler and Navier-Stokes equations*, Dept. of Mech. Engg., Ph. D. Thesis, Stanford University, Nov. 1988.
- [14] Johnson, C., Navert, U. and Pitkaranta, J., "Finite element methods for linear hyperbolic problems," *Comp. Meth. Appl. Mech. Engg.*, vol. 45, pp. 285-312, 1984.
- [15] Gummel, H. K., "A self-consistent iterative scheme for one-dimensional steady state transistor calculations," *IEEE Trans. Electron Devices*, vol. ED-11, pp. 455-465, 1964.
- [16] Jerome, J. W. and Shu, C., "Energy models for one-carrier transport in semiconductor devices," *ICASE report*, no. 91-78, 1991.



N. R. Aluru is currently a Doctoral student at Stanford university. He received his Bachelor of engineering degree from Birla Institute of Technology and Science (BITS), Pilani, India, in 1989 with honors and distinction. He is a recipient of several merit scholarships. He got a Master of Science degree from Rensselaer Polytechnic Institute, Troy, USA in 1991. His masters thesis involved three dimensional mesh generation employing octree based techniques. His areas of interest include semiconductor device simulation, parallel processing, finite element analysis, advanced numerical methods, computational mechanics, and large-scale computing.



Kincho H. Law is an associate professor of Civil Engineering at Stanford University. His research interests include computational mechanics, numerical methods, high performance computing, and analysis and design of large scale systems. He received his B.S. degree in Civil Engineering and his B.A. degree in Mathematics from the University of Hawaii in 1976 and his M.S. and Ph.D. degrees (both Civil Engineering) from Carnegie-Mellon University in 1979 and 1981, respectively. He taught at Rensselaer Polytechnic Institute for six years before joining Stanford University in 1988.



Peter M. Pinsky is an Associate Professor of Civil and Mechanical Engineering at Stanford University. His research is in the area of computational mechanics, specializing in the application of finite element methods for structural and solid mechanics. Pinsky received his B.S. from the University of Wales, Swansea, and his M.S. from the University of Toronto. He worked as a consulting structural engineer for six years before obtaining his Ph.D. in Civil Engineering from the University of California at Berkeley in 1983. He taught at Brown University for one year and joined the Stanford faculty in 1984.



Arthur Raefsky is an expert in parallel processing and has over fifteen years of experience in the development of numeric codes currently being used by scientists worldwide. He is known for his innovative solutions to complex engineering and physics problems and his integration of mathematics, computers and scientific disciplines. He received his B.S. (1971) in Physics from New York Institute of Technology. He received his M.S. (1974) and Ph.D. (1977) in Applied Mathematics and Computer Science from State University of New York at Binghamton. His thesis was concerned with design of software for the solution of very large systems of ordinary differential equations. Following graduate school, he was a postdoctoral fellow for three years at the California Institute of Technology's Seismological laboratory. He then worked concurrently as a senior research associate at Cal Tech and a member of the technical staff at Jet Propulsion Laboratory. Prior to joining CENTRIC, he was a senior research associate at Stanford's Applied Electronics Laboratory, working with Professor Robert Dutton solving problems in parallel processing and modeling of semiconductor devices. His research has involved the design of finite element software with focus on transportable software architectures for shared and local memory multi-processor computers. He has developed robust solution methods for large-scale finite element problems on parallel and vector computers. In particular, he has produced production level finite element software for the analysis of geophysical computational fluid dynamics problems that perform at high rates of efficiency for both the CRAY/YMP and INTEL/iPSC2 Hypercube. While at Cal Tech and JPL, he developed two and three dimensional non-Newtonian viscoelastic finite element analysis codes for modeling motion along earthquake faults and the thermal convective motions of the earth's mantle. These codes are used extensively in the national and international geophysical academic and research communities. He has authored or coauthored over 70 publications on software development and application of computational methods in engineering and geophysics. In addition, as a member of the Cal Tech Concurrent Computer project, he was one of the original developers of the Cal Tech Hypercube. In 1986 and 1988, he received the NASA Award for Outstanding Technological Innovation for his design of transportable software for multi-processor computer architectures.



Ronald Goossens is an expert in semiconductor physics and in the simulation of semiconductor processes and devices. He is known for his innovative solutions to complex engineering and physics problems and his integration of mathematics, computers and scientific disciplines. He received a triple B.S. (1974) in Physics, Mathematics and Astronomy, a double M.S. (1979) in Physics and Astronomy, and a Ph.D.

(1984) in Physics, all from Utrecht University in The Netherlands. He then joined the IC R & D Lab at Philips Research (1984) and worked for five years on various topics in relation to the development of their 1 Mbit SRAM. In 1989 he went to Stanford University as a liaison to the Center for Integrated Systems on behalf of Philips. In 1991 he joined Stanford University as a senior research associate to work with Prof. Dutton on the development of TCAD tools. Since then, he has consulted for various companies in the area of TCAD. Since July 1993, he is with National Semiconductor and is in charge of all process and device modeling for their Analog Division. His research has involved the design of submicron transistors, latchup analysis, compact transistor model development, hot-electron effects, as well as the design and application of TCAD tools. He has authored or coauthored over 30 publications. In 1992 he received an honorable mention in relation to the Intel Grand Challenge Computing Award for breakthrough innovations in the use of parallel computing to large-scale 3D semiconductor device simulation and its application to bipolar device scaling.



Robert W. Dutton is Professor of Electrical Engineering at Stanford University and Director of Research in the Center for Integrated systems. He received the B.S., M.S., and Ph.D. degrees from the University of California, Berkeley, in 1966, 1967, and 1970, respectively. He has held summer staff positions at Fairchild, Bell Telephone Laboratories, Hewlett-Packard, IBM Research, and Matsushita during 1967, 1973, 1975, 1977,

and 1988 respectively. His research interests focus on Integrated Circuit process, device, and circuit technologies—especially the use of Computer-Aided Design (CAD) and parallel computational methods. Dr. Dutton has published more than 200 journal articles and graduated more than four dozen doctorate students. He was Editor of the IEEE CAD Journal (1984–1986), winner of the 1987 IEEE J. J. Ebers Award, 1988 Guggenheim Fellowship to study in Japan and was elected to the National Academy of Engineering in 1991.

Numerical Solution of Two-Carrier Hydrodynamic Semiconductor Device Equations Employing a Stabilized Finite Element Method

N. R. Aluru, K. H. Law, A. Raefsky, P. M. Pinsky and R. W. Dutton

Integrated Circuits Laboratory

231-F, Applied Electronics Laboratory

Stanford University, Stanford, California 94309

Abstract

A space-time Galerkin/least-squares finite element method was presented in [1] for numerical simulation of single-carrier hydrodynamic semiconductor device equations. The single-carrier hydrodynamic device equations were shown to resemble the ideal gas equations and Galerkin/least-squares finite element method, originally developed for computational fluid dynamics equations [16], was extended to solve semiconductor device applications. In this paper, the space-time Galerkin/least-squares finite element method is further extended and generalized to solve two-carrier hydrodynamic device equations. The proposed formulation is based on a time-discontinuous Galerkin method, in which physical entropy variables are employed. A standard Galerkin finite element method is applied to the Poisson equation. Numerical simulations are performed on the coupled Poisson and the two-carrier hydrodynamic equations employing a staggered approach.

A mathematical analysis of the time-dependent multi-dimensional hydrodynamic model is performed to determine well-posed boundary conditions for electrical contacts. The number of boundary conditions that need to be specified for the hydrodynamic equations at inflow and outflow boundaries of the device are derived. Example boundary conditions that are based either on physical and/or mathematical basis are presented.

Stability of the numerical algorithms is addressed. The space-time Galerkin/least-squares finite element method and the standard Galerkin finite element method for the hydrodynamic and the Poisson equations, respectively, are shown to be stable. Specifically, a Clausius-Duhem inequality, a basic stability requirement, is derived for the hydrodynamic equations and the proposed numerical method automatically satisfies this stability requirement. Numerical simulations are performed on one and two dimensional two-carrier p-n diodes and the results demonstrate the effectiveness of the proposed numerical method.

Notation

α	particle identification; takes the value of 1 for electrons or 2 for holes
Γ	boundary of spatial domain
Γ_g	spatial boundary where essential boundary conditions are prescribed
Γ_h	spatial boundary where natural boundary conditions are prescribed
\mathcal{H}_α	generalized entropy function for the α^{th} particle
θ	dielectric permittivity
κ_α	heat-conductivity of the α^{th} particle
\mathcal{L}_α	differential operator for the α^{th} particle
$\lambda_{\alpha ij}$	eigenvalues of convective Jacobian matrix for the α^{th} particle
$\mu_{0\alpha}$	low-field mobility of the α^{th} particle
μ_α	specific chemical potential of the α^{th} particle
Ω	spatial domain
Ω_n^e	element spatial domain at the n^{th} time slab
ψ	electrostatic potential
$\bar{\psi}$	weighting function for the Poisson equation
ψ_{appl}	applied bias
ψ_b	built-in potential
$\tau_{GLS\alpha}$	intrinsic time scales matrix of the α^{th} particle employed in Galerkin/least-squares formulation
τ_1	electron life time
τ_2	hole life time
$\tau_{p\alpha}$	momentum relaxation time for the α^{th} particle
$\tau_{w\alpha}$	energy relaxation time for the α^{th} particle
γ_α	constant defining ratio of specific heats for the α^{th} particle
δ_{ij}	Kronecker delta; =1 for $i = j$ and 0 otherwise
$[]_{col}$	collision terms
$A_{\alpha i}$	convective Jacobian matrix of the α^{th} particle with respect to conservation variables in direction i
$\tilde{A}_{\alpha 0}$	Riemannian metric tensor for the α^{th} particle
$\tilde{A}_{\alpha i}$	convective Jacobian matrix of the α^{th} particle with respect to entropy variables in direction i

$\hat{A}_{\alpha i}$	convective Jacobian matrix of the α^{th} particle with respect to primitive variables in direction i
B_n	boundary of n^{th} space time slab
$B(\cdot, \cdot)_{\alpha n}$	left hand side operator for the weak form of the hydrodynamic equations for the α^{th} particle at time level n
$B_{GAL}(\cdot, \cdot)_{\alpha n}$	left hand side operator for the Galerkin form of the hydrodynamic equations for the α^{th} particle at time level n
$B_{GLS}(\cdot, \cdot)_{\alpha n}$	left hand side operator for the Galerkin/least-squares form of the hydrodynamic equations for the α^{th} particle at time level n
$B_p(\cdot, \cdot)$	left hand side operator for the Poisson equation
C_α	speed of sound for the α^{th} particle
D_α	vector of nonlinear boundary conditions for the α^{th} particle
E	electric field vector
E_i	electric field along direction i
F_α	source vector for the α^{th} particle
\hat{F}_α	source vector for the α^{th} particle when primitive variables are used
$F_{\alpha i}^c$	convective flux vector for the α^{th} particle in direction i
$F_{\alpha i}^h$	heat flux vector for the α^{th} particle in direction i
G	avalanche generation term (neglected in this paper)
I_n	n^{th} time interval
$K_{\alpha ij}$	diffusion matrix for the α^{th} particle with respect to conservation variables in directions i, j
$\tilde{K}_{\alpha ij}$	diffusion matrix for the α^{th} particle with respect to entropy variables in directions i, j
$\hat{K}_{\alpha ij}$	diffusion matrix for the α^{th} particle with respect to primitive variables in directions i, j
$L(\cdot)_{\alpha n}$	right hand side operator for the weak form of the hydrodynamic equations for the α^{th} particle at time level n
$L_{GAL}(\cdot)_{\alpha n}$	right hand side operator for the Galerkin form of the hydrodynamic equations for the α^{th} particle at time level n
$L_{GLS}(\cdot)_{\alpha n}$	right hand side operator for the Galerkin/least-squares form of the hydrody-

	dynamic equations for the α^{th} particle at time level n
$L_p(.)$	right hand side operator for the Poisson equation
$N_A^{(n)}$	finite element spatial shape function of node A for the n^{th} time slab
N_A^-	concentration of ionized acceptor
N_D^+	concentration of ionized donor
P_α	pressure of the α^{th} particle
Q_n	space time slab at time level n
Q_n^e	element space time slab at time level n
R_α	gas constant of the α^{th} particle
R, R_{SRH}, R_{AU}	recombination, Shockley-Read-Hall recombination and Auger recombination
T_0	lattice temperature
T_α	temperature of the α^{th} particle
U_α	conservative variable vector of the α^{th} particle
\hat{U}_α	primitive variable vector of the α^{th} particle
V_α	entropy variable vector of the α^{th} particle
c_α	carrier concentration of the α^{th} particle
c_1	concentration of electrons
c_2	concentration of holes
c_{int}	intrinsic carrier concentration
$c_{v\alpha}$	specific heat of the α^{th} particle at constant volume
e_α^{int}	internal energy of the α^{th} particle per unit mass
e_α^{kin}	kinetic energy of the α^{th} particle per unit mass
e_α^{tot}	total energy of the α^{th} particle per unit mass
$g_{\alpha i}$	prescribed boundary conditions of the α^{th} particle along direction i
k_b	Boltzmann constant
m_0	free electron mass
m_α	mass of the α^{th} particle
m_1	electron mass
m_2	hole mass
n_i	unit outward normal
$ndof$	number of degrees of freedom for hydrodynamic equations

$(nel)_n$	number of space-time finite elements at time level n
nsd	number of space dimensions
p_α	momentum density vector of the α^{th} particle
q_α	heat flux vector of the α^{th} particle
$q_{\alpha i}$	heat flux of the α^{th} particle along direction i
s_α	thermodynamic entropy of the α^{th} particle per unit mass
u_α	velocity vector of the α^{th} particle
$u_{\alpha i}$	velocity of the α^{th} particle along direction i
$v_{s\alpha}$	saturation velocity of the α^{th} particle
$v_{\alpha A}^{(n+1)}$	vector of nodal unknowns at node A for the hydrodynamic system of the α^{th} particle at time level $n + 1$
w_α	energy density of the α^{th} particle
$w_{\alpha A}^{(n+1)}$	vector of weighting functions at node A for the hydrodynamic system of the α^{th} particle at time level $n + 1$
$w_{0\alpha}$	equilibrium energy density of the α^{th} particle
X'	denotes the variation of X
$()^h$	denotes a finite element approximation
$()_0$	denotes a reference value

1 Introduction

The classical drift-diffusion (DD) equations for semiconductor device modeling assume a simple linear relationship between carrier velocity and the local electric field and negligible temperature gradients. The first assumption suppresses the velocity overshoot phenomena where the velocity can locally exceed the asymptotic limit placed by the DD model and the second assumption suppresses the carrier heating phenomena. With the scaling of silicon devices into deep submicron regimes, non-stationary phenomena such as velocity overshoot and carrier heating are becoming increasingly important to determine the characteristics of these devices. As a result, there has been a shift away from the commonly employed DD model and advanced transport models, such as the energy transport (ET) and the hydrodynamic (HD) models, have become increasingly popular. Both energy transport and hydrodynamic models can be derived from the Boltzmann Transport Equation (BTE) and the hydrodynamic model involves fewer assumptions compared to the energy transport model. In the hydrodynamic model, the carrier drift velocity is solved explicitly and this is needed for accurate description of the state-of-the-art devices. Hence the selection of the hydrodynamic model for semiconductor device simulation in this study.

The electrical current inside a material results from the transport of mobile charges called carriers. For semiconductors, after applying the energy band model to the periodic potentials of the crystal lattice, these carriers can be viewed as two types of oppositely-charged free particles moving in vacuum with modified effective mass and permittivity. The positively-charged carriers are called holes and the negatively-charged carriers are called electrons. Comprehensive semiconductor device simulation based on the hydrodynamic model involves solving a system of coupled electron hydrodynamic equations, hole hydrodynamic equations and the Poisson equation. This system is referred to as a two-carrier (involving both electrons and holes) hydrodynamic transport model. The device operation can be approximated by single-carrier (either electron or hole) in some simplified cases and numerical results based on the hydrodynamic model have been presented for single-carrier devices [8], [9], [24], [1], [2]. In [8], [9] and [24] finite difference and volume based schemes were employed. In our work [1], [2] a space-time Galerkin/least-squares (GLS) finite element scheme was employed. Finite element methods provide a more general framework than finite difference or volume based schemes, but are generally considered unsuitable for device applications [25]. The complex interaction between electrons and holes gives rise to solutions which vary several orders of magnitude within a few Angstroms. Robust numerical schemes are needed to guarantee stability, convergence and accuracy. In this paper the finite element numerical scheme presented in [1] is generalized to solve two-carrier hydrodynamic device equations. Our numerical results dispel the myth that finite elements are not suitable for semiconductor device simulation.

This paper addresses a number of numerical and mathematical issues related to the hydrodynamic model. First, the resemblance of the hydrodynamic equations to the ideal gas equations is exploited. The finite element numerical schemes developed by Hughes et al. [16] for compressible Euler and Navier-Stokes equations are extended to efficiently solve the coupled hydrodynamic equations. Second, the issue of boundary conditions for the hydrodynamic model is addressed. The number of boundary conditions to be specified for electrical contacts are derived and it is shown that the num-

ber of boundary conditions to be specified for the hydrodynamic model are different from those of the Euler and Navier-Stokes equations. Several sets of boundary conditions are proposed for subsonic/supersonic inflows/outflows. Practical difficulties in specifying well-posed conditions are addressed. Third, the stability of the proposed numerical schemes is established. Specifically, Clausius-Duhem inequalities are derived for the hydrodynamic device equations and the numerical scheme is shown to satisfy these inequalities.

This paper is organized as follows: Section 2 introduces the two-carrier hydrodynamic semiconductor device equations and the Poisson equation. Section 3 describes the assumptions employed in the hydrodynamic model, discusses the relationship to ideal gas equations, presents a conservation form on which the symmetrization procedures are developed, and introduces a finite element variational formulation. Section 4 presents theoretical results on boundary conditions. Section 5 discusses the GLS numerical scheme for hydrodynamic equations, establishes the stability and consistency of the numerical scheme. Section 6 presents a brief overview of the standard Galerkin finite element method for Poisson equations and establishes the stability and consistency of the method. Section 7 presents the solution scheme to solve the coupled two-carrier hydrodynamic and Poisson equations. Section 8 presents numerical results for one dimensional and two dimensional diodes and conclusions are presented in Section 9.

2 Semiconductor Equations

The motion of electrons and holes within a semiconductor can be best described by the integro-differential Boltzmann Transport Equation. Closed-form solution for this equation is not possible except for a few simple cases. The most successful approach to solve the BTE is by Monte Carlo simulation. An attractive alternative for semiconductor device simulation is to employ the hydrodynamic model. The hydrodynamic semiconductor device equations can be derived from the BTE by considering the first three moments, defining respectively, the particle continuity, conservation of momentum and energy [5] for the electrons and holes. The two systems of equations obtained from the first three moments of BTE can be summarized as follows:

$$\frac{\partial c_\alpha}{\partial t} + \nabla \cdot (c_\alpha \mathbf{u}_\alpha) = \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} \quad (1)$$

$$\frac{\partial \mathbf{p}_\alpha}{\partial t} + \mathbf{u}_\alpha (\nabla \cdot \mathbf{p}_\alpha) + (\mathbf{p}_\alpha \cdot \nabla) \mathbf{u}_\alpha = (-1)^\alpha \epsilon c_\alpha \mathbf{E} - \nabla (c_\alpha k_b T_\alpha) + \left[\frac{\partial \mathbf{p}_\alpha}{\partial t} \right]_{col} \quad (2)$$

$$\frac{\partial w_\alpha}{\partial t} + \nabla \cdot (\mathbf{u}_\alpha w_\alpha) = (-1)^\alpha \epsilon c_\alpha (\mathbf{u}_\alpha \cdot \mathbf{E}) - \nabla \cdot (\mathbf{u}_\alpha c_\alpha k_b T_\alpha) - \nabla \cdot \mathbf{q}_\alpha + \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} \quad (3)$$

for $\alpha = 1, 2$. Repeated index α does not imply summation. In Equations (1) - (3), c_α is the particle concentration; \mathbf{u}_α is the particle velocity vector, \mathbf{p}_α is the particle momentum density vector; T_α is the particle temperature; w_α is the particle energy density; \mathbf{q}_α is the particle heat flux vector; ϵ is the mag-

nitude of an elementary charge; k_b is the Boltzmann constant and $[]_{col}$ denotes the collision terms accounting for the particle-particle interactions, particle-lattice interactions, the transfer of energy between particle and lattice, and the generation and recombination process.

As noted above, Equations (1) - (3) represent two systems of equations corresponding to $\alpha = 1$ and $\alpha = 2$. We define the system with $\alpha = 1$ to be the equations governing the electrons, and the system with $\alpha = 2$ to be the equations governing the holes. In the sequel, Greek subscript, α , designates the electron and hole system according to the above stated convention and repeated Greek subscript does not imply summation. α is part of the variable symbol to emphasize the system to which it belongs.

The electron and hole concentrations are coupled to the electrostatic potential, ψ , by the Poisson equation. The Poisson equation, derived from Maxwells equations [25], is given by

$$\nabla \cdot (\theta E) = -\epsilon (c_1 - c_2 - N_D^+ + N_A^-) \quad (4)$$

where θ is the dielectric permittivity, N_D^+ is the concentration of the ionized donor, N_A^- is the concentration of ionized acceptor and E is the electric field vector. The electric field is related to the electrostatic potential by the equation

$$E = -\nabla \psi \quad (5)$$

3 Hydrodynamic model

3.1 Simplification and assumptions

Equations (1)-(5) represent an indeterminate system of equations as the number of unknowns are more than the number of equations. In order to facilitate a solution to the device model a few constitutive approximations need to be made. The carrier momentum density vector can be represented as

$$p_\alpha = m_\alpha c_\alpha u_\alpha \quad (6)$$

where m_α represents the particle mass. Note that $m_1 = 0.26m_0$ and $m_2 = 0.386m_0$, where m_0 is the free electron mass. The carrier energy density can be expressed as

$$w_\alpha = \frac{3}{2} c_\alpha k_b T_\alpha + \frac{1}{2} m_\alpha c_\alpha |u_\alpha|^2 \quad (7)$$

The heat conduction is assumed to be given by the Fourier law i.e.

$$q_\alpha = -\kappa_\alpha \nabla T_\alpha \quad (8)$$

The particle heat-conductivity κ_α is given by the Wiedemann-Franz law as

$$\kappa_{\alpha} = \left(\frac{5}{2} + \varsigma \right) \frac{\mu_{0\alpha} c_{\alpha} k_b^2 T_0}{\epsilon} \quad (9)$$

where $\mu_{0\alpha}$ is the particle low-field mobility and T_0 is the lattice temperature (which is assumed to be constant in this paper). ς is a parameter associated with the energy dependence of the momentum relaxation time. In this study, $\varsigma = -2$ is employed.

The collision term, $\left[\frac{\partial c_{\alpha}}{\partial t} \right]_{col}$, in Equation (1) describes the rate of change of particle concentration due to collisions. This term is neglected for single carrier devices, as in [1]. In the presence of both electrons and holes, this collision term has significant contribution to the transport equations and introduces coupling between the electron and hole transport systems. The collision term for the continuity equation describes the generation and recombination processes and has the following form

$$\left[\frac{\partial c_{\alpha}}{\partial t} \right]_{col} = G - R \quad (10)$$

where G is the avalanche generation term and R is the recombination term. The recombination term is a sum of Shockley-Read-Hall and Auger recombinations [25] i.e.

$$R = R_{SRH} + R_{AU} \quad (11)$$

The physical processes involved with the Auger recombination and the avalanche generation terms remain subjects of active investigation; these terms are not modeled in this study. The Shockley-Read-Hall recombination is given by

$$R_{SRH} = \frac{c_1 c_2 - c_{int}^2}{\tau_2 (c_1 + c_{int}) + \tau_1 (c_2 + c_{int})} \quad (12)$$

where c_{int} is the intrinsic carrier concentration for the silicon material, τ_1 is the electron life time and τ_2 is the hole life time and a value of 10^{-7} s is employed for both electron and hole lifetimes in this study.

The collision term, $\left[\frac{\partial p_{\alpha}}{\partial t} \right]_{col}$, in Equation (2) describes the particle rate of change of momentum due to collisions. This collision term can be treated by employing a relaxation time approximation [4] as

$$\left[\frac{\partial p_{\alpha}}{\partial t} \right]_{col} = -\frac{p_{\alpha}}{\tau_{p\alpha}} + \frac{p_{\alpha}}{c_{\alpha}} \left[\frac{\partial c_{\alpha}}{\partial t} \right]_{col} \quad (13)$$

The second term in the above equation accounts for the rate of change of momentum due to particle generation and recombination processes. The validity of this term is still a subject of active investigation. This term is included in our model as an option. The simulation results presented in this paper,

however, do not include this term. In Equation (13), $\tau_{p\alpha}$ denotes the momentum relaxation time given by

$$\tau_{p\alpha} = \frac{m_\alpha \mu_{0\alpha} T_0}{\epsilon T_\alpha} \quad (14)$$

The collision term, $\left[\frac{\partial w_\alpha}{\partial t} \right]_{col}$, in Equation (3) describes the particle rate of change of energy due to collisions. This collision term can also be treated by employing a relaxation time approximation as

$$\left[\frac{\partial w_\alpha}{\partial t} \right]_{col} = -\frac{(w_\alpha - w_{0\alpha})}{\tau_{w\alpha}} + \frac{w_\alpha}{c_\alpha} \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} \quad (15)$$

The second term in the above equation accounts for the rate of change of energy density due to particle generation and recombination processes. Similar to the discussion on the collision term for the momentum conservation equation, our numerical results do not include this term although it can be included as an option to our model easily. In Equation (15)

$$w_{0\alpha} = \frac{3}{2} c_\alpha k_b T_0 \quad (16)$$

denotes the equilibrium energy density and $\tau_{w\alpha}$ denotes the energy relaxation time expressed as

$$\tau_{w\alpha} = \frac{3}{2} \frac{\mu_{0\alpha}}{\epsilon v_{s\alpha}^2} \left(\frac{k_b T_\alpha T_0}{T_\alpha + T_0} \right) + \frac{\tau_{p\alpha}}{2} \quad (17)$$

and $v_{s\alpha}$ denotes the particle saturation velocity.

3.2 Relationship to Ideal gas equations

In [1] we have established the resemblance of the hydrodynamic semiconductor device equations for single carrier transport to the compressible Euler and Navier-Stokes equations of fluid dynamics. The result can be extended to the two carrier transport problem in a straight forward manner. Specifically, one can treat the electron and hole transport equations analogous to interacting flows with two different gas types. Formally, the resemblance to ideal gas equations can be stated as follows:

The α^{th} particle/carrier hydrodynamic transport equations, without neglecting the convective terms, represent the flow of an ideal gas with the particle gas constant $R_\alpha = k_b/m_\alpha$, the ratio of specific heats $\gamma_\alpha = 5/3$, pressure $P_\alpha = c_\alpha R_\alpha T_\alpha$ and the total energy per unit mass $e_\alpha^{tot} = 1.5 R_\alpha T_\alpha + 0.5 |u_\alpha|^2 = e_\alpha^{int} + e_\alpha^{kin}$, where e_α^{int} and e_α^{kin} denote the internal and kinetic energies per unit mass respectively. Furthermore, the α^{th} carrier transport equations resemble the compressible Euler equations with the addition of the heat conduction term, the collision terms and the electric

field terms which couple with the Poisson equation.

3.3 System Form and Symmetrization

The two-carrier hydrodynamic equations stated in Equations (1) - (3) can be put in the form of a system of equations as

$$U_{\alpha,i} + F_{\alpha i}^c = F_{\alpha i}^h + F_{\alpha} \quad (18)$$

where

$$U_{\alpha} = \begin{Bmatrix} U_{\alpha 1} \\ U_{\alpha 2} \\ U_{\alpha 3} \\ U_{\alpha 4} \\ U_{\alpha 5} \end{Bmatrix} = c_{\alpha} \begin{Bmatrix} 1 \\ u_{\alpha 1} \\ u_{\alpha 2} \\ u_{\alpha 3} \\ e_{\alpha}^{tot} \end{Bmatrix} \quad F_{\alpha i}^c = c_{\alpha} u_{\alpha i} \begin{Bmatrix} 1 \\ u_{\alpha 1} \\ u_{\alpha 2} \\ u_{\alpha 3} \\ e_{\alpha}^{tot} \end{Bmatrix} + P_{\alpha} \begin{Bmatrix} 0 \\ \delta_{1i} \\ \delta_{2i} \\ \delta_{3i} \\ u_{\alpha i} \end{Bmatrix} \quad (19)$$

$$F_{\alpha i}^h = \begin{Bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -q_{\alpha i} \end{Bmatrix} \quad (20)$$

$$F_{\alpha} = \begin{Bmatrix} -\frac{(c_1 c_2 - c_{int}^2)}{\tau_2 (c_1 + c_{int}) + \tau_1 (c_2 + c_{int})} \\ \frac{\epsilon c_{\alpha}}{m_{\alpha}} \left[(-1)^{\alpha} E_1 - \frac{u_{\alpha 1} T_{\alpha}}{\mu_{0\alpha} T_0} \right] \\ \frac{\epsilon c_{\alpha}}{m_{\alpha}} \left[(-1)^{\alpha} E_2 - \frac{u_{\alpha 2} T_{\alpha}}{\mu_{0\alpha} T_0} \right] \\ \frac{\epsilon c_{\alpha}}{m_{\alpha}} \left[(-1)^{\alpha} E_3 - \frac{u_{\alpha 3} T_{\alpha}}{\mu_{0\alpha} T_0} \right] \\ \frac{1}{m_{\alpha}} \left[(-1)^{\alpha} \epsilon c_{\alpha} u_{\alpha i} E_i - \frac{(c_{\alpha} m_{\alpha} e_{\alpha}^{tot} - \frac{3}{2} c_{\alpha} k_b T_0)}{\left(\frac{3 \mu_{0\alpha} k_b T_{\alpha} T_0}{2 \epsilon v_{s\alpha}^2 (T_{\alpha} + T_0)} + \frac{m_{\alpha} \mu_{0\alpha} T_0}{2 \epsilon T_{\alpha}} \right)} \right] \end{Bmatrix} \quad (21)$$

It is useful to rewrite Equation (18) in the quasi-linear form as

$$U_{\alpha,t} + A_{\alpha i} U_{\alpha,i} = (K_{\alpha ij} U_{\alpha,j})_{,i} + F_{\alpha} \quad (22)$$

where $A_{\alpha i} = F_{\alpha i, U_{\alpha}}^c$ and $K_{\alpha ij} U_{\alpha,j} = F_{\alpha i}^h$. The matrices $A_{\alpha i}$ do not possess the properties of symmetry or positiveness and, in general, are functions of U_{α} . In the following, a brief review is presented on the symmetrization techniques for Equation (22) as the finite element formulation based on a symmetrized form of Equation (22) can be shown to be unconditionally stable.

Symmetrization procedures for systems of form (22) have been investigated by Harten [13]. A generalized entropy function was proposed by Hughes et al. [17] for symmetrization of compressible Euler and Navier-Stokes equations. In [1] a generalized entropy function was employed for symmetrization of electron hydrodynamic transport equations. Since the form of the advection and diffusion matrix operators is similar for both electron and hole transport equations, a function similar to the one employed for electron hydrodynamic equations can also be employed for hole hydrodynamic equations. Employing generalized entropy functions of the form

$$\mathcal{H}_{\alpha} = -c_{\alpha} s_{\alpha} \quad (23)$$

where s_{α} is the thermodynamic entropy per unit mass, a symmetrized form for Equation (22) can be obtained as

$$\tilde{A}_{\alpha 0} V_{\alpha,t} + \tilde{A}_{\alpha i} V_{\alpha,i} = (\tilde{K}_{\alpha ij} V_{\alpha,j})_{,i} + F_{\alpha} \quad (24)$$

where the matrix operators $\tilde{A}_{\alpha i}$, $\tilde{K}_{\alpha ij}$ are symmetric and $\tilde{A}_{\alpha 0}$ is symmetric and positive definite. V_{α} are referred to as entropy variables for particle α and are defined as

$$V_{\alpha} = \frac{\partial \mathcal{H}_{\alpha}}{\partial U_{\alpha}} = \frac{1}{T_{\alpha}} \begin{bmatrix} \mu_{\alpha} - \frac{|u_{\alpha}|^2}{2} \\ u_{\alpha 1} \\ u_{\alpha 2} \\ u_{\alpha 3} \\ -1 \end{bmatrix} \quad (25)$$

where $\mu_{\alpha} = e_{\alpha}^{int} + \frac{P_{\alpha}}{c_{\alpha}} - T_{\alpha} s_{\alpha}$ is the particle specific chemical potential. The specific form of s_{α} is given by

$$s_{\alpha} = c_{v\alpha} \ln \left(\frac{P_{\alpha}}{P_{0\alpha}} \left(\frac{c_{\alpha}}{c_{0\alpha}} \right)^{-\gamma_{\alpha}} \right) + s_{0\alpha} \quad (26)$$

where $c_{v\alpha}$ is the particle specific heat at constant volume and the quantities with subscript "0" denote the reference quantities. The definitions of the symmetrized matrix operators have been given in [1]

for the electron hydrodynamic system. The matrix operators for hole hydrodynamic system can be defined in a similar manner by properly replacing the electron transport quantities with hole transport quantities.

4 Boundary Conditions

Well-posed boundary conditions play an important role in numerical simulations. Prescribing too many boundary conditions may preclude the existence of smooth solutions. Specifying too few boundary conditions, on the other hand may preclude uniqueness of the solution. Specification of improper number of boundary conditions can affect the convergence of the numerical schemes. Hence, it is important that one specifies the proper set of boundary conditions for numerical simulations. Well-posed boundary conditions for the classical DD model are well understood. The same set of boundary conditions, however, do not give well-posedness for the HD model. Thomann and Odeh [30] have shown that the boundary conditions based on the DD model are not sufficient for the HD model. While they have shown that additional boundary conditions are needed for the HD model, their analysis has been focused on the 2D hydrodynamic model and for subsonic flows.

Bova and Carey [6] have reported a study on boundary conditions for HD equations, taking advantage of the resemblance of HD equations to compressible Euler and Navier-Stokes equations. The number of boundary conditions that they have proposed are identical to those specified for Euler equations. In doing so they assumed that the diffusive effect of the heat flux term on the average energy is small on the boundaries; however, this assumption lacks a physical basis. As shall be shown in this paper, the proper number of boundary conditions that need to be specified for the HD equations are not identical to those of the Euler or Navier-Stokes equations. Well-posed boundary conditions for Euler and Navier-Stokes equations have been investigated by Strikwerda [29], Gustafson and Sundstrom [12], Olinger and Sundstrom [22], among others. The concepts developed in these studies are extended to derive well-posed boundary conditions for the HD equations. In this paper an analysis is performed on the general multi-dimensional (one, two and three dimensional) HD equations to include the heat flux term and to place no restriction on the type of flow, albeit subsonic or supersonic nature.

4.1 Primitive variable form

The two-carrier hydrodynamic equations discussed in this paper can be written in system form using primitive variables $(c_\alpha, u_\alpha, T_\alpha)$. The primitive variables are used to analyze the number of boundary conditions that need to be specified at the inflow and the outflow boundaries, that constitute a well-posed Initial Boundary Value Problem (IBVP). Using primitive variables, the conservation laws can be written using matrix-operators as

$$\frac{\partial \hat{U}_\alpha}{\partial t} = \hat{A}_{\alpha i} \frac{\partial \hat{U}_\alpha}{\partial x_i} + \hat{K}_{\alpha ij} \frac{\partial^2 \hat{U}_\alpha}{\partial x_i \partial x_j} + \hat{F}_\alpha \quad \alpha = 1, 2 \quad (27)$$

where \hat{U}_α denotes the primitive variables, $\hat{A}_{\alpha i}$ denotes the advection matrices, $\hat{K}_{\alpha ij}$ denotes the diffu-

sion matrices and \hat{F}_α denotes the source vector consisting of the collision and electric field terms. The explicit definitions of the advection matrices are given below with $\hat{U}_\alpha = \{T_\alpha, c_\alpha, u_\alpha\}^T$

$$\hat{A}_{\alpha 1} = \begin{bmatrix} -u_{\alpha 1} & 0 & -(\gamma_\alpha - 1)T_\alpha & 0 & 0 \\ 0 & -u_{\alpha 1} & -c_\alpha & 0 & 0 \\ -R_\alpha - \frac{R_\alpha T_\alpha}{c_\alpha} & -u_{\alpha 1} & 0 & 0 & 0 \\ 0 & 0 & 0 & -u_{\alpha 1} & 0 \\ 0 & 0 & 0 & 0 & -u_{\alpha 1} \end{bmatrix} \quad (28)$$

$$\hat{A}_{\alpha 2} = \begin{bmatrix} -u_{\alpha 2} & 0 & 0 & -(\gamma_\alpha - 1)T_\alpha & 0 \\ 0 & -u_{\alpha 2} & 0 & -c_\alpha & 0 \\ 0 & 0 & -u_{\alpha 2} & 0 & 0 \\ -R_\alpha - \frac{R_\alpha T_\alpha}{c_\alpha} & 0 & -u_{\alpha 2} & 0 & 0 \\ 0 & 0 & 0 & 0 & -u_{\alpha 2} \end{bmatrix} \quad (29)$$

$$\hat{A}_{\alpha 3} = \begin{bmatrix} -u_{\alpha 3} & 0 & 0 & 0 & -(\gamma_\alpha - 1)T_\alpha \\ 0 & -u_{\alpha 3} & 0 & 0 & -c_\alpha \\ 0 & 0 & -u_{\alpha 3} & 0 & 0 \\ 0 & 0 & 0 & -u_{\alpha 3} & 0 \\ -R_\alpha - \frac{R_\alpha T_\alpha}{c_\alpha} & 0 & 0 & 0 & -u_{\alpha 3} \end{bmatrix} \quad (30)$$

Note that $\hat{A}_{\alpha i}$ are square but non-symmetric matrices. Similarly, the diffusion matrices can be expressed as $\hat{K}_{\alpha ij} = \hat{K}_\alpha \delta_{ij}$ where δ_{ij} is the kronecker delta ($\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$) and

$$\hat{K}_\alpha = \begin{bmatrix} \frac{\hat{\kappa}_\alpha (\gamma_\alpha - 1)}{c_\alpha m_\alpha R_\alpha} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (31)$$

$\hat{K}_{\alpha ij}$ are rank-deficient matrices.

4.2 Conditions for well-posedness

The literature on well-posedness for incompletely parabolic problems dates back to 1970's. Strikwerda's thesis [29] on well-posed boundary conditions for incompletely parabolic problems addressed several issues related to the necessary and sufficient conditions for the PDE systems of form (27) to be well-posed. This work also paved way for a number of studies addressing boundary conditions for several physical problems. Of notable interest is the one by Gustafson and Sundstrom [12], addressing the issue of well-posed boundary conditions for equations of fluid dynamics and shallow water. By following the work in these two references, we extend the concepts to study the proper boundary conditions for the HD device equations, which can be considered as intermediary between Euler and Navier-Stokes (NS) equations. To derive the number of boundary conditions that need to be imposed at inflow and outflow boundaries, several results reported in references [29] and [12] are utilized. The main theorems and the definitions needed are briefly stated here; interested readers are referred to the references for the proofs.

Definition 1: Let $\hat{U}_{0\alpha}$ be the initial conditions to (27). The system (27) is said to be well-posed if there is a constant \hat{C}_α such that

$$\|\hat{U}_\alpha\| \leq \hat{C}_\alpha (\|\hat{U}_{0\alpha}\| + \|\hat{F}_\alpha\|) \quad (32)$$

Consider the incompletely parabolic system of partial differential equations given in (27) with constant coefficient matrices. The diffusion matrices $\hat{K}_{\alpha ij}$ are rank deficient with rank $1 < n$, where n is the order of the square matrices $\hat{A}_{\alpha i}$ and $\hat{K}_{\alpha ij}$. From Equation (31), it follows that

$$\hat{K}_{\alpha ij} = \begin{bmatrix} \hat{K}_{\alpha ij}^{(11)} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \quad (33)$$

where $\hat{K}_{\alpha ij}^{(11)} = \left[\frac{\hat{k}_\alpha (\gamma_\alpha - 1)}{c_\alpha m_\alpha R_\alpha} \right]_{1 \times 1}$. $\hat{A}_{\alpha i}$ is partitioned as

$$\hat{A}_{\alpha i} = \begin{bmatrix} \hat{A}_{\alpha i}^{(11)} & \hat{A}_{\alpha i}^{(12)} \\ \hat{A}_{\alpha i}^{(21)} & \hat{A}_{\alpha i}^{(22)} \end{bmatrix} \quad (34)$$

\hat{U}_α is also partitioned accordingly as $\hat{U}_\alpha = [\hat{U}_{\alpha l}, \hat{U}_{\alpha ll}]^T$ where $\hat{U}_{\alpha l} = T_\alpha$ and $\hat{U}_{\alpha ll} = \{c_\alpha, u_\alpha\}^T$.

Theorem 1 (Strikwerda [29] and Gustafson et. al [12]): System (27) is said to be well posed, if the system

$$\frac{\partial \hat{U}_{\alpha l}}{\partial t} = \hat{K}_{\alpha ij}^{(11)} \frac{\partial^2 \hat{U}_{\alpha l}}{\partial x_i \partial x_j} \quad (35)$$

is parabolic and that the system

$$\frac{\partial \hat{U}_{all}}{\partial t} = \hat{A}_{\alpha i}^{(22)} \frac{\partial \hat{U}_{all}}{\partial x_i} \quad (36)$$

is strictly hyperbolic.

Theorem 2 (Strikwerda [29]): Consider the initial boundary value problem for the system (27) on a half space; i.e. $x_1 \geq 0$ and $-\infty < x_2, x_3 < \infty$ with constant coefficients. For the system (27) to be well-posed the number of independent boundary conditions is given by $r + p$, where r is the rank of $\hat{K}_{\alpha 11}$ and p is the number of negative eigenvalues of $\hat{A}_{\alpha 1}^{(22)}$.

Theorem 3 (Strikwerda [29]): Suppose the system (27) is approximated by a set of frozen coefficient matrices. If the approximated system to (27) is well-posed, then system (27) is well-posed.

Remarks:

- i) Gustafson and Sundstrom [12] have shown that the definition given for well-posedness in Theorem 1 is not sufficient. They illustrated the problem using examples where the conditions stated in Theorem 1 are satisfied, but the solution has an exponential growth rate. However, such exponential growth rates cannot be obtained for symmetrizable incompletely parabolic systems. Since the NS and HD equations can be symmetrized, Theorem 1 is a sufficient condition for well-posedness.
- ii) Using the result in Theorem 2, analysis will be performed for an inflow boundary parallel to the y-axis (or an electrical contact parallel to y-axis). The analysis and results apply analogously to inflow boundaries parallel to x- or z-axis.
- iii) With Theorem 3, the examination of well-posed boundary conditions can be restricted to constant coefficient systems, instead of the more general quasi-linear system of equations.

4.3 Number of independent boundary conditions for HD equations

The theorems cited above can be directly applied to determine the number of independent boundary conditions for the HD equations. In the following, the analysis is performed on the equations for the general three-dimensional problem, and the results are analogously applicable for one- and two-dimensional problems. From the matrix definitions given in Equations (28)-(30), it is clear that the rank of the diffusion matrix $\hat{K}_{\alpha 11}$ is one and the submatrix $\hat{A}_{\alpha 1}^{(22)}$ of the advection matrix $\hat{A}_{\alpha 1}$ is given as

$$\hat{A}_{\alpha 1}^{(22)} = \begin{bmatrix} -u_{\alpha 1} & -c_{\alpha} & 0 & 0 \\ -\frac{R_{\alpha} T_{\alpha}}{c_{\alpha}} & -u_{\alpha 1} & 0 & 0 \\ 0 & 0 & -u_{\alpha 1} & 0 \\ 0 & 0 & 0 & -u_{\alpha 1} \end{bmatrix} \quad (37)$$

According to Theorem 2, the number of boundary conditions can be determined by finding the number of negative eigenvalues of the above matrix. The four eigenvalues of $\hat{A}_{\alpha 1}^{(22)}$ are

$$\begin{aligned}\lambda_{\alpha 1} &= \lambda_{\alpha 2} = -u_{\alpha 1} \\ \lambda_{\alpha 3} &= -u_{\alpha 1} + C_{\alpha} \\ \lambda_{\alpha 4} &= -u_{\alpha 1} - C_{\alpha}\end{aligned}\tag{38}$$

where $C_{\alpha} = \sqrt{R_{\alpha} T_{\alpha}}$ is the speed of sound. From (38), the number of boundary conditions can be derived by classifying the inflow and outflow as either subsonic ($|u_{\alpha 1}| < C_{\alpha}$) or supersonic ($|u_{\alpha 1}| > C_{\alpha}$) flow:

1. Subsonic inflow ($C_{\alpha} > u_{\alpha 1} > 0$): In this case three of the eigenvalues ($\lambda_{\alpha 1}, \lambda_{\alpha 2}, \lambda_{\alpha 4}$) of $\hat{A}_{\alpha 1}^{(22)}$ are negative. Thus a total of 4 boundary conditions are needed for the inflow to ensure well-posedness (For the Euler and NS equations four and five boundary conditions are needed, respectively, for the inflow).
2. Subsonic outflow ($0 > u_{\alpha 1} > -C_{\alpha}$): In this case there is only one negative eigenvalue ($\lambda_{\alpha 4}$) in $\hat{A}_{\alpha 1}^{(22)}$. Therefore, a total of two boundary conditions is needed for the outflow to ensure well-posedness of the system (For the Euler and NS equations, one and four boundary conditions are needed, respectively, for the outflow).
3. Supersonic inflow ($u_{\alpha 1} > C_{\alpha} > 0$): In this case all four eigenvalues of $\hat{A}_{\alpha 1}^{(22)}$ are negative. We thus need to specify five boundary conditions at the inflow for a well-posed system (The Euler and NS equations also require five boundary conditions).
4. Supersonic outflow ($0 > -C_{\alpha} > u_{\alpha 1}$): In this case all eigenvalues of $\hat{A}_{\alpha 1}^{(22)}$ are positive and we need to specify just one boundary condition at the outflow to ensure well-posedness of the system (For the Euler and NS equations, we need zero and four, boundary conditions, respectively, for the outflow).

Table 1 summarizes the number of independent boundary conditions for one-, two- and three dimensional flows for the Euler, Navier-Stokes and HD equations. In general we can express the number of boundary conditions in terms of the number of primitive variables (i.e. the degree of freedom *ndof* per each node) as tabulated in Table 2. Note that $ndof = nsd + 2$, where *nsd* is the number of space dimensions equal to 1, 2, 3 for 1D, 2D and 3D problems respectively.

4.4 Specification of boundary conditions

The classical energy method can be applied to show well-posedness for symmetrizable incompletely parabolic systems. In this approach, energy growth expressions are derived by considering the variational forms for the frozen coefficient system of equations (Equations (24) or (27)). These expressions have been derived for Euler and Navier-Stokes equations in [12] and for the hydrodynamic device equations in [3]. In these references it was shown that to obtain boundedness of the solution at all times, the boundary integrals contained in the energy growth expression need to be positive i.e.

Table 1: Number of independent boundary conditions

Type of flow		Euler	NS	HD
one dimensional flow	<i>subsonic inflow</i>	2	3	2
	<i>subsonic outflow</i>	1	2	2
	<i>supersonic inflow</i>	3	3	3
	<i>supersonic outflow</i>	0	2	1
two dimensional flow	<i>subsonic inflow</i>	3	4	3
	<i>subsonic outflow</i>	1	3	2
	<i>supersonic inflow</i>	4	4	4
	<i>supersonic outflow</i>	0	3	1
three dimensional flow	<i>subsonic inflow</i>	4	5	4
	<i>subsonic outflow</i>	1	4	2
	<i>supersonic inflow</i>	5	5	5
	<i>supersonic outflow</i>	0	4	1

$$\int_{\Gamma} V_{\alpha}^T \tilde{A}_{\alpha 1} V'_{\alpha} d\Gamma + 2 \int_{\Gamma} V_{\alpha}^T \tilde{K}_{\alpha 11} \frac{\partial V'_{\alpha}}{\partial x_1} d\Gamma \geq 0 \quad (39)$$

where V'_{α} denotes the variation of V_{α} . The definition for V'_{α} is given in Equation (40).

Table 2: Summary of independent boundary conditions for 1-, 2-, and 3D flows

Type of flow	Euler	NS	HD
<i>subsonic inflow</i>	<i>ndof-1</i>	<i>ndof</i>	<i>ndof-1</i>
<i>subsonic outflow</i>	1	<i>ndof-1</i>	2
<i>supersonic inflow</i>	<i>ndof</i>	<i>ndof</i>	<i>ndof</i>
<i>supersonic outflow</i>	0	<i>ndof-1</i>	1

$$V'_\alpha = \begin{bmatrix} -\frac{u_{\alpha i}}{T_\alpha} u'_{\alpha i} + \frac{|u_\alpha|^2}{2T_\alpha^2} T'_\alpha + R_\alpha \frac{c'_\alpha}{c_\alpha} - c_{v\alpha} \frac{T'_\alpha}{T_\alpha} \\ (T_\alpha u'_{\alpha 1} - u_{\alpha 1} T'_\alpha) / T_\alpha^2 \\ (T_\alpha u'_{\alpha 2} - u_{\alpha 2} T'_\alpha) / T_\alpha^2 \\ (T_\alpha u'_{\alpha 3} - u_{\alpha 3} T'_\alpha) / T_\alpha^2 \\ T'_\alpha / T_\alpha^2 \end{bmatrix} \quad (40)$$

Substituting the definitions for V'_α , $\tilde{A}_{\alpha 1}$ and $\tilde{K}_{\alpha 11}$, Equation (39) can be rewritten as

$$\begin{aligned} \frac{1}{T_\alpha} \left[-c_\alpha u_{\alpha 1} \left(\sum_{i=1}^{nsd} u'^2_{\alpha i} + \frac{R_\alpha T_\alpha}{(\gamma_\alpha - 1)} \left(\frac{T'_\alpha}{T_\alpha} \right)^2 + R_\alpha T_\alpha \left(\frac{c'_\alpha}{c_\alpha} \right)^2 \right) - 2R_\alpha T_\alpha c'_\alpha u'_{\alpha 1} \right] + \\ \frac{1}{T_\alpha} \left[-2c_\alpha R_\alpha T'_\alpha u'_{\alpha 1} + 2 \frac{\hat{\kappa}_\alpha}{m_\alpha} T'_\alpha T_\alpha^{-1} \frac{\partial}{\partial x} (T'_\alpha) \right] \geq 0 \end{aligned} \quad (41)$$

The boundary conditions for HD equations are imposed by satisfying the positivity condition specified in Equation (41). In the following, we will consider each of the four cases discussed before, i.e. subsonic/supersonic inflow and subsonic/supersonic outflow, and derive a set(s) of boundary conditions and show that these boundary conditions satisfy the inequality (41).

4.4.1 Subsonic inflow ($C_\alpha > u_{\alpha 1} > 0$)

From table 2 we need to specify 2, 3 and 4 boundary conditions for 1D, 2D and 3D, respectively. One set of possible boundary conditions are summarized below

1D: $c_\alpha u_{\alpha 1} = g_{\alpha 1}$ and $T_\alpha = g_{\alpha 2}$

2D: $c_\alpha u_{\alpha 1} = g_{\alpha 1}$, $u_{\alpha 2} = g_{\alpha 2}$ and $T_\alpha = g_{\alpha 3}$

3D: $c_\alpha u_{\alpha 1} = g_{\alpha 1}$, $u_{\alpha 2} = g_{\alpha 2}$, $u_{\alpha 3} = g_{\alpha 3}$ and $T_\alpha = g_{\alpha 4}$

where $g_{\alpha i}$ denotes a prescribed value for the quantity to be specified. In the following, it will be verified that the boundary conditions indeed satisfy the inequality (41). The prescribed boundary conditions would mean $u'_{\alpha 2} = u'_{\alpha 3} = T'_\alpha = 0$. Substituting these in Equation (41) would make the left hand side (*lhs*) of the inequality (41) as

$$lhs = -c_\alpha u_{\alpha 1} \left(u_{\alpha 1}'^2 + R_\alpha T_\alpha \left(\frac{c'_\alpha}{c_\alpha} \right)^2 \right) - 2R_\alpha T_\alpha u'_{\alpha 1} c'_\alpha \quad (42)$$

The boundary condition $c_\alpha u_{\alpha 1} = g_{\alpha 1}$ gives $\frac{c'_\alpha}{c_\alpha} = -\frac{u'_{\alpha 1}}{u_{\alpha 1}}$. Substituting this condition into Equation (42), we get

$$lhs = \frac{u_{\alpha 1}'^2}{g_{\alpha 1}} (-u_{\alpha 1}^2 + C_\alpha^2) > 0 \quad (43)$$

since the flow is subsonic. The boundary conditions for 1D and 2D cases can be verified in a similar manner.

A second set of boundary conditions that can be specified for subsonic inflow stems from Schottky barriers. In this type of boundary condition the normal component of current is related to the concentration. For electrons and holes, this condition is given as

$$-c_\alpha u_{\alpha 1} = v_{th} (c_\alpha - c_{0\alpha}) \quad (44)$$

where v_{th} is the thermionic velocity and $c_{0\alpha}$ is the equilibrium concentration. Using this condition, the second set of boundary conditions can be summarized as follows:

1D: $u_{\alpha 1} = -v_{th} \left(1 - \frac{c_{0\alpha}}{c_\alpha} \right)$ and $T_\alpha = g_{\alpha 2}$

2D: $u_{\alpha 1} = -v_{th} \left(1 - \frac{c_{0\alpha}}{c_\alpha} \right)$, $u_{\alpha 2} = g_{\alpha 2}$ and $T_\alpha = g_{\alpha 3}$

3D: $u_{\alpha 1} = -v_{th} \left(1 - \frac{c_{0\alpha}}{c_\alpha} \right)$, $u_{\alpha 2} = g_{\alpha 2}$, $u_{\alpha 3} = g_{\alpha 3}$ and $T_\alpha = g_{\alpha 4}$

For these boundary conditions, it can be shown that the inequality in Equation (41) would be satisfied for the following condition

$$0 \leq u_{\alpha 1} \leq 2 \left(\frac{C_\alpha^2 g_\alpha}{g_\alpha^2 + C_\alpha^2} \right) \quad (45)$$

where $g_\alpha = \frac{v_{th} c_{0\alpha}}{c_\alpha}$. The first set of boundary conditions are harder to implement for device examples as the quantity $c_\alpha u_{\alpha 1}$ is not generally known.

It is to be observed that the prescribed c_α , T_α and tangential components of velocity (for multi-dimensional flows) are not well-posed boundary conditions, even though these are the commonly employed boundary conditions for device simulation. We do not suggest that the boundary conditions discussed above (and hereafter) exhaust all possible sets of boundary conditions. For instance, in the case of a high level injection of a diode, none of the above sets of boundary conditions seem to be appropriate. Development of a set of proper boundary conditions for such a device remains a subject for further investigation.

4.4.2 Subsonic outflow ($0 > u_{\alpha 1} > -C_\alpha$)

For subsonic outflow, regardless of the space dimension of the problem, two boundary conditions need to be specified. Inequality (41) can be satisfied by choosing any of the following three sets of boundary conditions.

1: $c_\alpha = g_{\alpha 1}$ and $T_\alpha = g_{\alpha 2}$

2: $u_{\alpha 1} = -v_{th} \left(1 - \frac{c_{0\alpha}}{c_\alpha} \right)$ and $T_\alpha = g_{\alpha 2}$

3: $u_{\alpha 1} = g_{\alpha 3}$ and $\frac{\partial T_\alpha}{\partial x} = g_{\alpha 4}$

In semiconductor device simulation, the inflow velocity $u_{\alpha 1}$ is typically not known. So the first two sets of boundary conditions are more appropriate compared to the third set. For the first set of boundary conditions the inequality is satisfied, i.e

$$-c_\alpha u_{\alpha 1} \left(\sum_{i=1}^{nsd} u_{\alpha i}^2 + \frac{R_\alpha T_\alpha}{(\gamma_\alpha - 1)} \left(\frac{T_\alpha}{T_\alpha} \right)^2 + R_\alpha T_\alpha \left(\frac{c'_\alpha}{c_\alpha} \right)^2 \right) > 0 \quad (46)$$

since $u_{\alpha 1} < 0$ and the quantity inside the parenthesis is positive. In the second set of boundary conditions (again based on Schottky barrier), a velocity boundary condition similar to the one suggested for subsonic inflow is employed. In this case the inequality takes the form

$$-c_\alpha u_{\alpha 1} \left(\sum_{i=1}^{nsd} u_{\alpha i}^2 + \frac{R_\alpha T_\alpha}{(\gamma_\alpha - 1)} \left(\frac{T_\alpha}{T_\alpha} \right)^2 + R_\alpha T_\alpha \left(\frac{c'_\alpha}{c_\alpha} \right)^2 \right) + \frac{2R_\alpha T_\alpha v_{th} c_{0\alpha}}{c_\alpha^2} c'^2_\alpha \geq 0 \quad (47)$$

since $u_{\alpha 1} < 0$. Note that for this set of boundary conditions no limit is placed on the inflow velocity $u_{\alpha 1}$.

Commonly employed boundary conditions for 2D simulations (assume the contact placement is parallel to x-axis) are $c_\alpha = g_{\alpha 1}$, $u_{\alpha 2} = 0$ and $T_\alpha = g_{\alpha 3}$. Based on the above development this set of boundary conditions appears to be an over specification.

4.4.3 Supersonic inflow ($u_{\alpha 1} > C_{\alpha} > 0$)

For supersonic inflow one needs to specify 3, 4 and 5 boundary conditions for 1D, 2D and 3D problems respectively. The number of conditions would imply that all the basic nodal variables be specified. So the following set of boundary conditions can be specified

$$T_{\alpha} = g_{\alpha nsd+2}, c_{\alpha} = g_{\alpha;1}, \text{ and } u_{\alpha;i} = g_{\alpha;i+1} \text{ for } i = 1, nsd$$

The boundary conditions mentioned here pose an interesting physical question. As noted earlier, the inflow velocity is typically not known i.e. $u_{\alpha 1}$ is not known. However, since the flow is supersonic we may impose that the inflow velocity cannot be greater than the saturation velocity. Alternatively, any other set of boundary conditions that satisfies the inequality (41) are also applicable. For the boundary conditions specified above, the left-hand-side of Equation (41) is equal to zero. It should be mentioned that in semiconductor device simulation, supersonic inflow boundaries are rarely encountered.

4.4.4 Supersonic outflow ($0 > -C_{\alpha} > u_{\alpha 1}$)

Independent of the space dimension, only one boundary condition needs to be specified for this case.

Valid boundary conditions include setting $\frac{\partial T_{\alpha}}{\partial x} = g_{\alpha 1}$ or $T_{\alpha} = g_{\alpha 2}$. In this case the inequality takes the form

$$lhs = -c_{\alpha} u_{\alpha 1} \left(\sum_{i=1}^{nsd} u_{\alpha i}^2 + \frac{R_{\alpha} T_{\alpha}}{(\gamma_{\alpha} - 1)} \left(\frac{T'_{\alpha}}{T_{\alpha}} \right)^2 + R_{\alpha} T_{\alpha} \left(\frac{c'_{\alpha}}{c_{\alpha}} \right)^2 \right) - 2R_{\alpha} T_{\alpha} c'_{\alpha} u'_{\alpha 1} - 2c_{\alpha} R_{\alpha} T'_{\alpha} u'_{\alpha 1} \quad (48)$$

This equation can be rewritten as:

$$\begin{aligned} lhs = & -c_{\alpha} u_{\alpha 1} \left(\sum_{i=2}^{nsd} u_{\alpha i}^2 \right) - \frac{u_{\alpha 1} c_{\alpha} R_{\alpha} T_{\alpha}}{\gamma_{\alpha} (\gamma_{\alpha} - 1)} \left[\left(\frac{T'_{\alpha}}{T_{\alpha}} - (\gamma_{\alpha} - 1) \frac{c'_{\alpha}}{c_{\alpha}} \right)^2 \right] \\ & + \frac{1}{2} (-u_{\alpha 1} - C_{\alpha}) c_{\alpha} \left[u'_{\alpha 1} + \sqrt{\frac{R_{\alpha} T_{\alpha}}{\gamma_{\alpha}}} \left(\frac{T'_{\alpha}}{T_{\alpha}} + \frac{c'_{\alpha}}{c_{\alpha}} \right) \right]^2 \\ & + \frac{1}{2} (-u_{\alpha 1} + C_{\alpha}) c_{\alpha} \left[u'_{\alpha 1} - \sqrt{\frac{R_{\alpha} T_{\alpha}}{\gamma_{\alpha}}} \left(\frac{T'_{\alpha}}{T_{\alpha}} + \frac{c'_{\alpha}}{c_{\alpha}} \right) \right]^2 \end{aligned} \quad (49)$$

In this case $u_{\alpha 1} < 0$ and both $(-u_{\alpha 1} - C_{\alpha})$ and $(-u_{\alpha 1} + C_{\alpha})$ are positive; hence the inequality (41) is satisfied.

Remarks:

- i) The examples presented for inflow and outflow boundaries and subsonic/supersonic cases are representatives of the possible sets of well-posed boundary conditions for the HD system. The boundary conditions discussed have either physical or mathematical basis and can easily be implemented.

- ii) Mixed type of boundary conditions (involving the quantity and its derivative) are among the feasible sets of boundary conditions. Reference [12] has some examples on this type for Euler and Navier-Stokes equations. Mixed type of boundary conditions are not presented here since they are usually more difficult to implement.
- iii) In practice, simulations are performed without verifying the well-posedness of the boundary conditions. If stable numerical schemes are employed, exponential growth in the solution can be avoided. However, where possible it is recommended that well-posed boundary conditions be specified to avoid steep gradients in the solution and to ensure the convergence behavior of the numerical scheme.

5 Numerical Scheme for two-carrier hydrodynamic equations

The most common numerical schemes employed for semiconductor device simulation are finite difference and finite volume based schemes. See [23], [25] for an overview of finite difference or volume based schemes for drift diffusion equations, [7] for the extension of these schemes to energy transport equations and [9], [8] for the application of difference based schemes to hydrodynamic equations.

Finite element methods have not been attempted with much success to device simulation [25] as the standard Galerkin finite element method exhibits spurious oscillations when the exact solution contains steep layers. Hughes and Brooks developed a Streamline Upwind Petrov-Galerkin (SUPG) [15] finite element method which can resolve steep layers in the exact solution efficiently. Sharma and Carey [28] implemented this SUPG finite element formulation for the traditional drift diffusion equations. Hughes et al. [26], [27], [17], [18], [16] generalized the SUPG finite element formulation to Galerkin/least-squares finite element formulation and successfully applied it to compressible and incompressible behavior of fluids. In [1], a Galerkin/least-squares finite element formulation is applied to treat the single carrier hydrodynamic semiconductor device equations. In Galerkin/least-squares finite element formulation terms of a least-squares type are added to the variational equation obtained from the Galerkin method. These least-squares terms vanish when the exact solution is obtained, thus making it a consistent method. GLS is a higher order accurate method with good stability properties. The temporal behavior of the HD equations is discretized using a discontinuous Galerkin method in time [19]. This discretization consists of a constant-in-time approximation, which leads to an inexpensive and highly stable first-order time-accurate algorithm, ideal for steady problems.

In this section the details on extending the Galerkin/least-squares formulation to two-carrier hydrodynamic device equations are presented.

5.1 Variational forms for the hydrodynamic equations

Let the variational functional spaces S_n and ϑ_n both consist of continuous functions with square integrable first derivatives within each space-time slab. The solution space S_n is the set of all such functions satisfying the essential boundary conditions. While the weighting-function space, ϑ_n , is made up of functions whose value is zero where essential boundary conditions are specified i.e.

$$\begin{aligned} S_n &= \{V_\alpha \mid V_\alpha \in H^1(Q_n), D_\alpha(V_\alpha) = g_\alpha(t) \text{ on } B_n\} \\ \mathfrak{V}_n &= \{W_\alpha \mid W_\alpha \in H^1(Q_n), D'_\alpha(W_\alpha) = 0 \text{ on } B_n\} \end{aligned} \quad (50)$$

where $Q_n = \Omega \times I_n$ is the space-time slab with boundary $B_n = \Gamma \times I_n$. D_α and D'_α denote the non-linear boundary condition operators for the α^{th} carrier and g_α denotes the vector of prescribed boundary conditions. Ω denotes the multi-dimensional spatial domain with boundary Γ and $I_n =]t_n, t_{n+1}[$ denotes the n th time interval with t_n and t_{n+1} as the n th and $(n+1)$ -th time levels, respectively. Before stating the weak form, it is useful to introduce the following notation:

$$(W_\alpha, V_\alpha)_{Q_n} = \int_{Q_n} (W_\alpha \cdot V_\alpha) dQ \quad (51)$$

$$(W_\alpha, V_\alpha)_\Omega = \int_\Omega (W_\alpha \cdot V_\alpha) d\Omega \quad (52)$$

$$a(W_\alpha, V_\alpha)_{Q_n} = \int_{Q_n} (W_{\alpha,i} \cdot \tilde{K}_{\alpha ij} V_{\alpha,j}) dQ \quad (53)$$

$$(W_\alpha, V_\alpha)_{B_n} = \int_{B_n} (W_\alpha \cdot V_\alpha) n_i dB \quad (54)$$

$$(W_\alpha, V_\alpha)_{Q_n^\varepsilon} = \sum_{\varepsilon=1}^{(n_{el})_n} \int_{Q_n^\varepsilon} (W_\alpha \cdot V_\alpha) dQ \quad (55)$$

In Equation (55), $(n_{el})_n$ denotes the number of space-time finite elements at time level n , $Q_n^\varepsilon = \Omega_n^\varepsilon \times I_n$ denotes the domain of element interior, and n_i denotes the unit outward normal. Note that the operators defined in Equations (51) - (55) are symmetric i.e. $(W_\alpha, V_\alpha)_\Omega = (V_\alpha, W_\alpha)_\Omega$.

The weak form can be stated as follows: Within each Q_n , $n = 0, \dots, N-1$, find $V_\alpha \in S_n$ such that for all $W_\alpha \in \mathfrak{V}_n$ the following variational equation is satisfied:

$$B(W_\alpha, V_\alpha)_{\alpha n} = L(W_\alpha)_{\alpha n} \quad (56)$$

where

$$\begin{aligned} B(W_\alpha, V_\alpha)_{\alpha n} &= ((-W_{\alpha,t}), U_\alpha(V_\alpha))_{Q_n} - ((W_{\alpha,i}), F_{\alpha i}^c(V_\alpha))_{Q_n} + a(W_\alpha, V_\alpha)_{Q_n} - \\ & (W_\alpha, F_\alpha(V_\alpha))_{Q_n} + (W_\alpha(t_{n+1}^-), U_\alpha(V_\alpha(t_{n+1}^-)))_\Omega + (W_\alpha, F_{\alpha i}^c(V_\alpha) - F_{\alpha i}^h(V_\alpha))_{B_n} \end{aligned} \quad (57)$$

and

$$L(W_\alpha)_{\alpha n} = (W_\alpha(t_n^+), U_\alpha(V_\alpha(t_n^+)))_\Omega \quad (58)$$

Equations (57) and (58) are obtained by multiplying Equation (18) with the weighting function and performing integration by parts. It is to be observed that the operator B in Equation (56) is non-symmetric.

Let S_n^h and \mathfrak{V}_n^h be the finite-dimensional approximations to S_n and \mathfrak{V}_n , respectively. The time-discontinuous Galerkin formulation can be written as follows:

Within each Q_n , $n = 0, \dots, N-1$, find $V_\alpha^h \in S_n^h$ such that for all $W_\alpha^h \in \mathfrak{V}_n^h$ the following variational equation is satisfied:

$$B_{GAL}(W_\alpha^h, V_\alpha^h)_{\alpha n} = L_{GAL}(W_\alpha^h)_{\alpha n} \quad (59)$$

where

$$\begin{aligned} B_{GAL}(W_\alpha^h, V_\alpha^h)_{\alpha n} &= B(W_\alpha^h, V_\alpha^h)_{\alpha n} \\ L_{GAL}(W_\alpha^h)_{\alpha n} &= (W_\alpha^h(t_n^+), U_\alpha(V_\alpha^h(t_n^-))) \end{aligned} \quad (60)$$

A jump condition term of the following form

$$\int_\Omega W_\alpha^h(t_n^+) \cdot [\![U_\alpha(V_\alpha^h(t_n))]\!] d\Omega \quad (61)$$

is added to the variational equation to enforce weak initial conditions for each space-time slab. The term

$$[\![U_\alpha(t_n)]\!] = U_\alpha(t_n^+) - U_\alpha(t_n^-) \quad (62)$$

denotes the jump in time of U_α in the time slab.

The Galerkin finite element formulation summarized in Equation (59) possess poor stability properties when the global solution has steep gradients. Spurious oscillations are often observed in the vicinity of steep layers. In the following a time-discontinuous Galerkin/least-squares formulation is developed which possess improved stability properties as well as robustness.

5.2 Time-Discontinuous Galerkin/least-squares formulation

The space-time Galerkin/least-squares finite element formulation for the α^{th} carrier hydrodynamic transport equations can be stated as follows:

Within each Q_n , $n = 0, \dots, N-1$, find $V_\alpha^h \in S_n^h$ such that for all $W_\alpha^h \in \mathfrak{V}_n^h$ the following variational equation is satisfied:

$$B_{GLS}(W_\alpha^h, V_\alpha^h)_{\alpha n} = L_{GLS}(W_\alpha^h)_{\alpha n} \quad (63)$$

where

$$B_{GLS}(W_\alpha^h, V_\alpha^h)_{\alpha n} = B_{GAL}(W_\alpha^h, V_\alpha^h)_{\alpha n} + (\mathcal{L}_\alpha W_\alpha^h, \tau_{GLS\alpha} \mathcal{L}_\alpha V_\alpha^h)_{Q_n^\Sigma} \quad (64)$$

$$L_{GLS}(W_\alpha^h)_{\alpha n} = L_{GAL}(W_\alpha^h)_{\alpha n} \quad (65)$$

The stability emanates from the addition of a least-squares term to the Galerkin formulation

$$\sum_{e=1}^{(n_e)_n} \int_{Q_n^e} (\mathcal{L}_\alpha W_\alpha^h) \cdot \tau_{GLS\alpha} (\mathcal{L}_\alpha V_\alpha^h) dQ \quad (66)$$

The least-squares term is proportional to the residual and therefore only contributes to regions where the Galerkin method fails to resolve the transport of carriers. The governing differential operator, \mathcal{L}_α is given by

$$\mathcal{L}_\alpha = \tilde{A}_{\alpha 0} \frac{\partial}{\partial t} + \tilde{A}_{\alpha i} \frac{\partial}{\partial x_i} - \frac{\partial}{\partial x_i} (\tilde{K}_{\alpha ij} \frac{\partial}{\partial x_j}) + \tilde{C}_\alpha \quad (67)$$

where \tilde{C}_α is a non-unique operator and is defined as

$$F_\alpha = -\tilde{C}_\alpha V_\alpha \quad (68)$$

$\tau_{GLS\alpha}$ is an $n_{dof} \times n_{dof}$ symmetric positive-semidefinite matrix of intrinsic time scales. This is discussed in greater detail in the next sub-section.

Finite element discretization:

The finite element interpolation is introduced

$$V_\alpha^h = \sum_{A=1}^{(n_{np})_n} N_A^{(n)}(x) v_{\alpha A}^{(n+1)} \quad (69)$$

where $v_{\alpha A}^{(n+1)}$ are α^{th} carriers $n_{dof} \times 1$ nodal unknowns, $N_A^{(n)}(x)$ is the matrix of shape functions for n^{th} space-time slab and $(n_{np})_n$ is the number of finite element nodes for the n^{th} space-time slab. As in the Galerkin finite element method, the weighting functions are interpolated using the same functions $N_A^{(n)}(x)$ i.e.

$$W_\alpha^h = \sum_{A=1}^{(n_{np})_n} N_A^{(n)}(x) w_{\alpha A}^{(n+1)} \quad (70)$$

Substituting the finite element interpolants, Equations (69) and (70), into the Galerkin/least-squares variational equation, Equation (63), a nonlinear systems of equations is obtained

$$G_{\alpha}(v_{\alpha}; v_1^{(n)}; v_2^{(n)}) = 0 \quad (71)$$

Equation (71) means that the nonlinear algebraic equations to be solved at time-level $(n+1)$ for the α^{th} carrier, G_{α} , are a function of the α^{th} carrier entropy variables at time-level $(n+1)$ and the electron and hole entropy variables at time-level n , $v_1^{(n)}$ and $v_2^{(n)}$, respectively. The nonlinear system of equations can be solved by linearizing Equation (71) with respect to the unknown variables v_{α} and applying a time-stepping algorithm in the format of a predictor multi-corrector algorithm [26], [1].

5.2.1 Design of intrinsic time scales matrix: τ_{GLS}

A design of the time scales matrix τ_{GLS} for nonlinear hydrodynamic equations is very complicated. Generally, simpler one-dimensional scalar equations are used as model problems and the results obtained from the analysis of the one-dimensional model problems are extended to multi-dimensional systems. Definitions provided for multi-dimensional systems are not necessarily optimal. Hughes et al. [18] have examined such approach to the modeling of several fluid flow problems and they showed that excellent results can be obtained when one-dimensional results are extended to non-linear multi-dimensional problems. Employing a similar approach, we consider the following scalar one-dimensional advection-diffusion equation with source term

$$u\phi_{,x} = k\phi_{,xx} + c_s\phi \quad (72)$$

Following the conditions given by Shakib [26], a τ for the scalar advection-diffusion equation with source term can be written as

$$\tau_s = (c_s^2 + (2\frac{u}{h})^2 + (\frac{4k}{h^2})^2)^{-0.5} \quad (73)$$

where the subscript s denotes the scalar equation and h is the mesh size parameter. Equation (73) can be rewritten as

$$\tau_s = \frac{h}{2u} \cdot \sqrt{\frac{\alpha_s^2}{1 + \alpha_s^2}} \cdot \left(1 + c_s^2 \left(\frac{h}{2u}\right)^2 \frac{\alpha_s^2}{1 + \alpha_s^2}\right)^{-0.5} \quad (74)$$

where $\alpha_s = \frac{uh}{2k}$. In Equation (74), the first term in the product of three terms can be considered as the design of τ for advection limit case (i.e. in the absence of diffusion and source) and the next two terms can be viewed as the corrections for the presence of diffusion and source terms respectively. More optimal definitions of τ can be derived for Equation (72). However, they are more expensive and the gain is often little. The result obtained in Equation (74) can be generalized to the system of equations as discussed in the following:

Consider a constant-coefficient one-dimensional system of equations in the hydrodynamic form i.e.

$$U_{,t} + AU_{,x} = KU_{,xx} + F \quad (75)$$

Employing a change of variables, a symmetric system of equations can be obtained

$$\tilde{A}_0 V_{,t} + \tilde{A} V_{,x} = \tilde{K} V_{,xx} + \tilde{F} \quad (76)$$

\tilde{A}_0 is constant matrix and can be expressed in a product form as $\tilde{A}_0 = LL^T$. Defining $X = L^T V$, Equation (76) can be transformed into a new system of equations

$$X_{,t} + \bar{A} X_{,x} = \bar{K} X_{,xx} + \bar{F} \quad (77)$$

where $\bar{A} = L^{-1} \tilde{A} L^{-T}$, $\bar{K} = L^{-1} \tilde{K} L^{-T}$ and $\bar{F} = L^{-1} \tilde{F}$. The eigenvalues for the three systems of Equations, (75) - (77), are identical. If we denote the eigenvectors to be Ψ , Φ and v for the Equations (75), (76) and (77), respectively, then the following relation holds

$$\Psi = \tilde{A}_0 \Phi = L(L^T \Phi) = L\Upsilon \quad (78)$$

Defining, $X = \Upsilon \chi$, where $\Upsilon = [v_1, \dots, v_m]$ Equation (77) becomes

$$\Upsilon \chi_{,t} + \bar{A} \Upsilon \chi_{,x} = \bar{K} \Upsilon \chi_{,xx} + \bar{F} \quad (79)$$

where $\bar{A} = \Upsilon \Lambda \Upsilon^{-1}$ and $\Lambda = \text{diag} [\lambda_1, \dots, \lambda_m]$. Multiplying Equation (79) by Υ^T , from the left, one obtains

$$\chi_{,t} + \Lambda \chi_{,x} = \Upsilon^T \bar{K} \Upsilon \chi_{,xx} + \Upsilon^T \bar{F} \quad (80)$$

The similarity transformation discussed above diagonalizes only the \bar{A} matrix but not the diffusion and source matrices. More general transformation procedures can be considered to diagonalize more than one matrix at a time, but the procedures are more expensive. The term τ defined using the above procedure is generally sufficient to obtain a stable and robust finite element method.

The i th scalar component of Equation (80) can be written as

$$\chi_{i,t} + \lambda_i \chi_{i,x} = k_i \chi_{i,xx} + c_{si} \chi_i \quad (81)$$

where $k_i = \Upsilon_i^T \bar{K} \Upsilon_i = \Phi_i^T \tilde{K} \Phi_i$ and $c_{si} = -\Phi_i^T \tilde{C} \Phi_i$. Equation (81) is similar to the scalar advection-diffusion equation considered in Equation (72) and the two equations are in fact identical for steady state problems. A τ_{si} (subscript si denotes the i^{th} scalar equation) can then be defined for Equation (81) (analogous to Equation (73)). The intrinsic time scales matrix can now be defined by considering a Galerkin/least-squares formulation for Equation (77) and diagonalizing the variational equations using the transformation procedure discussed in [18]. This procedure leads to the definition of τ as

$$\tau_{GLS} = \Phi \text{diag} (\tau_{s1}, \dots, \tau_{si}, \dots, \tau_{sm}) \Phi^T \quad (82)$$

where τ_{si} is the definition for scalar equation given in Equation (81). For two-carrier hydrodynamic equations, Equation (82) can be generalized as

$$\tau_{GLS\alpha} = \Phi_{\alpha} \text{diag}(\tau_{GLS\alpha 1}, \dots, \tau_{GLS\alpha m}) \Phi_{\alpha}^T \quad (83)$$

5.2.2 Consistency

The consistency of the Galerkin/least-squares formulation, Equation (63), with the strong form of the boundary value problem may be verified by replacing V_{α}^h by V_{α} i.e.

$$B_{GLS}(W_{\alpha}^h, V_{\alpha})_{n\alpha} - L_{GLS}(W_{\alpha}^h)_{n\alpha} = 0 \quad (84)$$

Substituting the expressions for B_{GLS} and L_{GLS} (Equation (59)) into Equation (84) and integrating by parts, we obtain

$$\begin{aligned} & \int_{Q_n} W_{\alpha}^h \cdot [U_{\alpha,i}(V_{\alpha}) + F_{\alpha i,i}^c(V_{\alpha}) - F_{\alpha i,i}^h(V_{\alpha}) - F_{\alpha}(V_{\alpha})] dQ + \\ & \int_{\Omega} W_{\alpha}^h(t_n^+) \cdot [\llbracket U_{\alpha}(V_{\alpha}(t_n)) \rrbracket] d\Omega + \int_{Q_n} (\mathcal{L}_{\alpha} W_{\alpha}^h) \cdot \tau_{\alpha}(\mathcal{L}_{\alpha} V_{\alpha}) dQ = 0 \end{aligned} \quad (85)$$

Since V_{α} is the exact solution and is smooth, the residual and the jump term are exactly zero i.e.

$$\mathcal{L}_{\alpha} V_{\alpha} = 0 \quad (86)$$

$$[\llbracket U_{\alpha}(V_{\alpha}(t_n)) \rrbracket] = 0 \quad (87)$$

Since W_{α}^h is arbitrary, Equation (85) can be rewritten as

$$U_{\alpha,i}(V_{\alpha}) + F_{\alpha i,i}^c(V) - F_{\alpha i,i}^h(V) - F_{\alpha}(V) = 0 \quad (88)$$

which is the strong form of the problem stated in Equation (18).

5.2.3 Entropy production: stability analysis

In this section the hydrodynamic conservation laws are analyzed for stability. Stability of the numerical algorithms is vital for numerical calculations. It is a well known fact that the Clausius-Duhem inequality provides the conditions for physical stability of the system under consideration. It is crucial that the numerical algorithms obey these stability conditions. In the following, it will be established that the numerical algorithms discussed in this paper obey these stability conditions.

Clausius-Duhem inequality

In non-equilibrium thermodynamics, the balance equation for entropy reveals that the entropy of a volume element changes with time for two reasons: (1) entropy flows into the volume element and

(2) there is an entropy source due to irreversible phenomena inside the volume element. The entropy source is always a non-negative quantity, since entropy can only be created, never destroyed. For reversible transformations the entropy source vanishes. This is the local formulation of the second law of thermodynamics. By combining the second law of thermodynamics with the macroscopic laws of conservation of mass, momentum and energy an expression for the rate of change of the local entropy can be obtained [21].

The conservation laws contain a number of quantities such as the diffusion flows, the heat flow and the pressure tensor, which are related to the transport of mass, momentum and energy. The entropy source may then be calculated if one makes use of the Gibbs relation which connects the rate of change of entropy in each mass element to the rate of change of energy and the rates of change in composition. The Gibbs equation relating the entropy to the other properties of the system is given by

$$T_\alpha ds_\alpha = de_\alpha^{int} + P_\alpha dv_\alpha \quad (89)$$

In the following, a stability condition will be derived by appropriately modifying the conservation laws.

The equation for the conservation of momentum can be modified to obtain a balance equation for the creation of kinetic energy. Multiplying the momentum conservation equation by a velocity component $u_{\alpha i}$ and summing over all i the balance equation is obtained as

$$c_\alpha \frac{\partial}{\partial t} \left(\frac{|u_\alpha|^2}{2} \right) + c_\alpha u_{\alpha i} \frac{\partial}{\partial x_i} \left(\frac{|u_\alpha|^2}{2} \right) + u_{\alpha i} \frac{\partial P_\alpha}{\partial x_i} = (-1)^\alpha \frac{\epsilon}{m_\alpha} c_\alpha u_{\alpha i} E_i - |u_\alpha|^2 \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \quad (90)$$

Notice that the equation for conservation of particle number is utilized to obtain (90). In Equation (90), $\left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col}$ denotes the i th component of the collision term for the momentum equation. Subtracting the kinetic energy expression (90) from the energy conservation equation gives

$$\begin{aligned} c_\alpha \frac{\partial e_\alpha^{int}}{\partial t} + c_\alpha u_{\alpha i} \frac{\partial e_\alpha^{int}}{\partial x_i} + P_\alpha \frac{\partial u_{\alpha i}}{\partial x_i} + \left(e_\alpha^{int} - \frac{|u_\alpha|^2}{2} \right) \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \\ - \frac{1}{m_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} = - \frac{\partial q_{\alpha i}}{\partial x_i} \end{aligned} \quad (91)$$

Equation (91) can be rewritten as

$$\begin{aligned} c_\alpha \left(\frac{\partial e_\alpha^{int}}{\partial t} + P_\alpha \frac{\partial v_\alpha}{\partial t} \right) + c_\alpha u_{\alpha i} \left(\frac{\partial e_\alpha^{int}}{\partial x_i} + P_\alpha \frac{\partial v_\alpha}{\partial x_i} \right) + \left(h_\alpha - \frac{|u_\alpha|^2}{2} \right) \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \\ - \frac{1}{m_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} = - \frac{\partial q_{\alpha i}}{\partial x_i} \end{aligned} \quad (92)$$

where $v_\alpha = 1/c_\alpha$ is the specific volume and $h_\alpha = e_\alpha^{int} + P_\alpha v_\alpha$ is the specific enthalpy.

Using Gibbs relation, Equation (92) can be rewritten as

$$T_\alpha c_\alpha \frac{\partial s_\alpha}{\partial t} + T_\alpha c_\alpha u_{\alpha i} \frac{\partial s_\alpha}{\partial x_i} + \left(h_\alpha - \frac{|u_\alpha|^2}{2} \right) \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} - \frac{1}{m_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} = - \frac{\partial q_{\alpha i}}{\partial x_i} \quad (93)$$

Considering the Fourier heat conduction law $q_{\alpha i} = -\hat{\kappa}_\alpha T_{\alpha, i}$ and the expansion

$$- \frac{\partial q_{\alpha i}}{\partial x_i} = \frac{\hat{\kappa}_\alpha}{T_\alpha} (\nabla T_\alpha)^2 - T_\alpha (q_{\alpha i}/T_\alpha)_{,i} \quad (94)$$

Equation (93) can be written as

$$\begin{aligned} c_\alpha \frac{\partial s_\alpha}{\partial t} + c_\alpha u_{\alpha i} s_{\alpha, i} + (q_{\alpha i}/T_\alpha)_{,i} + \left(\frac{h_\alpha - \frac{|u_\alpha|^2}{2}}{T_\alpha} \right) \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \\ - \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} = \frac{\hat{\kappa}_\alpha}{T_\alpha^2} (\nabla T_\alpha)^2 \end{aligned} \quad (95)$$

For our intended purpose, Equation (95) can be rewritten as

$$\begin{aligned} (c_\alpha s_\alpha)_{,t} + (c_\alpha u_{\alpha i} s_\alpha)_{,i} + (q_{\alpha i}/T_\alpha)_{,i} + \left(\frac{h_\alpha - \frac{|u_\alpha|^2}{2}}{T_\alpha} - s_\alpha \right) \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \\ - \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} = \frac{\hat{\kappa}_\alpha}{T_\alpha^2} (\nabla T_\alpha)^2 \end{aligned} \quad (96)$$

Observing that the quantity on the right-hand side of Equation (96) is positive, an inequality of the following form can be obtained

$$\begin{aligned} (c_\alpha s_\alpha)_{,t} + (c_\alpha u_{\alpha i} s_\alpha)_{,i} + (q_{\alpha i}/T_\alpha)_{,i} + \left(\frac{h_\alpha - \frac{|u_\alpha|^2}{2}}{T_\alpha} - s_\alpha \right) \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \\ - \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} \geq 0 \end{aligned} \quad (97)$$

Equation (97) is the statement for the production of entropy and serves as the basic stability condition for the hydrodynamic device equations.

Remarks:

- i) Entropy production is governed by the Clausius-Duhem inequality. Clausius-Duhem inequality is also referred to as the second law of thermodynamics in some references (see [21] for a discussion). Thus, Equation (97) is referred to as the local form of the Clausius-Duhem inequality for the conservation laws governing hydrodynamic semiconductor device equations.
- ii) Clausius-Duhem inequalities can be derived for problems governed by the conservation laws of mass, momentum and energy and have been derived for the Euler and Navier-Stokes equations [17].
- iii) It is important to note the contribution of collision terms to entropy production Equation (97). Note also that the electric field terms do not contribute to entropy production.
- iv) The Clausius-Duhem inequality is also the physical stability condition for the conservation laws. Numerical formulations should not violate the Clausius-Duhem inequality.

Significance of entropy variables

The choice of variables employed to solve the set of conservation laws can play a significant role in the quality of numerical results. While any meaningful variables can be used to solve the conservation laws, we were motivated to use the entropy variables partially due to their success in producing superior results when applied to compressible Euler and Navier-Stokes equations [27]. Importantly, the use of entropy variables leads to a global statement of stability. A stability result is obtained by dotting the symmetrized system with V_α (vector of entropy variables):

$$V_\alpha \cdot (\tilde{A}_{\alpha 0} V_{\alpha,t} + \tilde{A}_{\alpha i} V_{\alpha,i} - (\tilde{K}_{\alpha ij} V_{\alpha,j})_{,i} - F_\alpha) = 0 \quad (98)$$

Noting that (these results can be obtained directly from the definition of coefficient matrices obtained using the entropy variables)

$$\begin{aligned} V_\alpha \cdot (\tilde{A}_{\alpha i} V_{\alpha,i}) &= (\mathcal{H}_\alpha u_{\alpha i})_{,i} \\ V_\alpha \cdot \tilde{F}_{\alpha,i}^h &= \frac{q_{\alpha i}}{T_\alpha} \\ V_\alpha \cdot F_\alpha &= \frac{V_{\alpha 1}}{T_\alpha} \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} - \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} \end{aligned} \quad (99)$$

Equation (98) can be rewritten as

$$\begin{aligned} \mathcal{H}_{\alpha,t} + (\mathcal{H}_\alpha u_{\alpha i})_{,i} - (q_{\alpha i}/T_\alpha)_{,i} - V_{\alpha 1} \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} - \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \\ + \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} = -\nabla V_\alpha \cdot \tilde{K}_\alpha \nabla V_\alpha \end{aligned} \quad (100)$$

Substituting $\mathcal{H}_\alpha = -c_\alpha s_\alpha$ in Equation (100), we obtain

$$\begin{aligned} (c_\alpha s_\alpha)_{,t} + (c_\alpha u_{\alpha i} s_\alpha)_{,i} + (q_{\alpha i}/T_\alpha)_{,i} + \left(\frac{h_\alpha - \frac{|u_\alpha|^2}{2}}{T_\alpha} - s_\alpha \right) \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} \\ - \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} \geq 0 \end{aligned} \quad (101)$$

Equation (101) is identical to the Clausius-Duhem inequality, Equation (97). That is, the Galerkin/least-squares finite element solution based on entropy variables automatically inherits the entropy production property of the hydrodynamic device equations.

Integrating Equation (100) over Q one obtains

$$\begin{aligned} \int_\Omega [\mathcal{H}_\alpha(T) - \mathcal{H}_\alpha(0)] d\Omega + \int_P \left[\mathcal{H}_\alpha u_{\alpha n} - \frac{q_{\alpha n}}{T_\alpha} \right] dP \\ - \int_Q \left\{ \frac{V_{\alpha 1}}{T_\alpha} \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} - \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} \right\} dQ = - \int_Q \nabla V_\alpha \cdot \tilde{K}_\alpha \nabla V_\alpha dQ \end{aligned} \quad (102)$$

Observing that the term on the right-hand side is non positive, Equation (102) can be expressed in an inequality form as

$$\begin{aligned} \int_\Omega [\mathcal{H}_\alpha(T) - \mathcal{H}_\alpha(0)] d\Omega + \int_P \left[\mathcal{H}_\alpha u_{\alpha n} - \frac{q_{\alpha n}}{T_\alpha} \right] dP \\ - \int_Q \left\{ \frac{V_{\alpha 1}}{T_\alpha} \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} + \frac{u_{\alpha i}}{m_\alpha T_\alpha} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} - \frac{1}{m_\alpha T_\alpha} \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} \right\} dQ \leq 0 \end{aligned} \quad (103)$$

Stability Result from Galerkin/least-squares variational form

In this section it will be shown that the numerical algorithms discussed in this paper obey the Clausius-Duhem inequality and hence are stable formulations. Consider the statement of the finite element space-time weighted residual formulation given in Equation (63).

Substituting V_α^h for W_α^h in Equation (63) and summing over all the time slabs i.e.

$$\sum_{n=0}^{N-1} (B_{GLS}(V^h, V^h)_{\alpha n} - L_{GLS}(V^h)_{\alpha n}) = 0 \quad (104)$$

$$\begin{aligned}
 & \sum_{n=0}^{N-1} \int_{Q_n} [-V_{\alpha,i}^h \cdot U_{\alpha}(V_{\alpha}^h) - V_{\alpha,i}^h \cdot F_{\alpha i}^c(V_{\alpha}^h) + V_{\alpha,i}^h \cdot \tilde{K}_{\alpha ij}(V_{\alpha}^h) V_{\alpha,j}^h - V_{\alpha}^h \cdot F_{\alpha}(V_{\alpha}^h)] dQ \\
 & + \sum_{n=0}^{N-1} \int_{\Omega} [V_{\alpha}^h(t_{n+1}^-) \cdot U_{\alpha}(V_{\alpha}^h(t_{n+1}^-)) - V_{\alpha}^h(t_n^+) \cdot U_{\alpha}(V_{\alpha}^h(t_n^+))] d\Omega \\
 & + \sum_{n=0}^{N-1} \int_{P_n} [V_{\alpha}^h \cdot (F_{\alpha i}^c(V_{\alpha}^h) - F_{\alpha i}^h(V_{\alpha}^h)) n_i] dP + \sum_{n=0}^{N-1} \int_{Q_n^c} \mathcal{L}_{\alpha} W_{\alpha}^h \cdot \tau_{GLS\alpha} \mathcal{L}_{\alpha} V_{\alpha}^h dQ = 0
 \end{aligned} \tag{105}$$

Furthermore, note that

$$\begin{aligned}
 & - \int_{Q_n} V_{\alpha,i}^h \cdot F_{\alpha i}^c(V_{\alpha}^h) dQ = - \int_{P_n} V_{\alpha}^h F_{\alpha i}^c n_i dP + \int_{Q_n} V_{\alpha}^h F_{\alpha i}^c dQ \\
 & \int_{Q_n} -V_{\alpha,i}^h \cdot U_{\alpha}(V_{\alpha}^h) dQ = \int_{Q_n} V_{\alpha}^h \cdot U_{\alpha,i}(V_{\alpha}^h) dQ - \int_{\Omega} [V_{\alpha}^h(t_{n+1}^-) \cdot U_{\alpha}^h(t_{n+1}^-) - V_{\alpha}^h(t_n^+) \cdot U_{\alpha}^h(t_n^+)] d\Omega
 \end{aligned} \tag{106}$$

Using Equations (99) and (106), Equation (105) can be rewritten as

$$\begin{aligned}
 & \sum_{n=0}^{N-1} \int_{Q_n} [-\mathcal{H}_{\alpha,i}^h - (\mathcal{H}_{\alpha}^h u_{\alpha i}^h)_{,i} + \nabla V_{\alpha}^h \cdot \tilde{K}_{\alpha ij}(V_{\alpha}^h) \nabla V_{\alpha}^h] dQ \\
 & - \sum_{n=0}^{N-1} \int_{Q_n} \left\{ \frac{V_{\alpha 1}^h}{T_{\alpha}^h} \left[\frac{\partial c_{\alpha}}{\partial t} \right]_{col} + \frac{u_{\alpha i}^h}{m_{\alpha} T_{\alpha}^h} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} - \frac{1}{m_{\alpha} T_{\alpha}^h} \left[\frac{\partial w_{\alpha}}{\partial t} \right]_{col} \right\} dQ \\
 & + \sum_{n=0}^{N-1} \int_{\Omega} [V_{\alpha}^h(t_n^+) \cdot (U_{\alpha}^h(t_n^+) - U_{\alpha}^h(t_n^-))] d\Omega - \sum_{n=0}^{N-1} \int_{P_n} \frac{q_{\alpha n}^h}{T_{\alpha}^h} dP + \sum_{n=0}^{N-1} \int_{Q_n^c} \mathcal{L}_{\alpha} W_{\alpha}^h \cdot \tau_{GLS\alpha} \mathcal{L}_{\alpha} V_{\alpha}^h dQ = 0
 \end{aligned} \tag{107}$$

where $\mathcal{H}_{\alpha}^h = \mathcal{H}_{\alpha}(U_{\alpha}(V_{\alpha}^h))$, $T_{\alpha}^h = T_{\alpha}(V_{\alpha}^h)$, $q_{\alpha n}^h = q_{\alpha i}(V_{\alpha}^h) n_i$ and $u_{\alpha i}^h = u_{\alpha i}(V_{\alpha}^h)$.

Defining

$$\|s\|_Q^2 = \int_Q s dQ \tag{108}$$

$$a_{\alpha n} = \int_{\Omega} V_{\alpha}^h(t_n^+) [U_{\alpha}(V_{\alpha}^h(t_n))] d\Omega - \int_{\Omega} (\mathcal{H}_{\alpha}^h(t_n^+) - \mathcal{H}_{\alpha}^h(t_n^-)) d\Omega \tag{109}$$

and noting that

$$\int_Q \mathcal{H}_{\alpha}^h dQ = \int_{\Omega} [\mathcal{H}_{\alpha}^h(T^-) - \mathcal{H}_{\alpha}^h(0^+)] d\Omega - \sum_{n=1}^{N-1} \int_{\Omega} [\mathcal{H}_{\alpha}^h(t_n^+) - \mathcal{H}_{\alpha}^h(t_n^-)] d\Omega \tag{110}$$

Equation (107) becomes

$$\begin{aligned}
 & \int_{\Omega} [\mathcal{H}_{\alpha}^h(T^-) - \mathcal{H}_{\alpha}^h(0^+)] d\Omega + \int_P \left(\mathcal{H}_{\alpha}^h u_{\alpha n}^h - \frac{q_{\alpha n}^h}{T_{\alpha}^h} \right) dP \\
 & - \int_Q \left\{ \frac{V_{\alpha 1}^h}{T_{\alpha}^h} \left[\frac{\partial c_{\alpha}}{\partial t} \right]_{col} + \frac{u_{\alpha i}^h}{m_{\alpha} T_{\alpha}^h} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} - \frac{1}{m_{\alpha} T_{\alpha}^h} \left[\frac{\partial w_{\alpha}}{\partial t} \right]_{col} \right\} dQ = \\
 & - \left\| \tilde{K}_{\alpha}^{0.5} \nabla V_{\alpha}^h \right\|_Q^2 - \left\| \tau_{GLS\alpha}^{0.5} \mathcal{L}_{\alpha} V_{\alpha}^h \right\|_Q^2 - \sum_{n=0}^{N-1} a_{\alpha n}
 \end{aligned} \tag{111}$$

In Equation (111), all the quantities on the right hand side are negative (see Appendix for the proof that $a_{\alpha n}$ is positive). Therefore Equation (111) can be rewritten as

$$\begin{aligned}
 & \int_{\Omega} [\mathcal{H}_{\alpha}^h(T^-) - \mathcal{H}_{\alpha}^h(0^+)] d\Omega + \int_P \left(\mathcal{H}_{\alpha}^h u_{\alpha n}^h - \frac{q_{\alpha n}^h}{T_{\alpha}^h} \right) dP \\
 & - \int_Q \left\{ \frac{V_{\alpha 1}^h}{T_{\alpha}^h} \left[\frac{\partial c_{\alpha}}{\partial t} \right]_{col} + \frac{u_{\alpha i}^h}{m_{\alpha} T_{\alpha}^h} \left[\frac{\partial p_{\alpha i}}{\partial t} \right]_{col} - \frac{1}{m_{\alpha} T_{\alpha}^h} \left[\frac{\partial w_{\alpha}}{\partial t} \right]_{col} \right\} dQ \leq 0
 \end{aligned} \tag{112}$$

Equation (112) is the exact analog of the entropy production inequality derived earlier in Equation (103). Hence we conclude that our numerical formulations conform with Clausius-Duhem inequality and are entropy stable. In Equation (111), the second and third terms on the right-hand side are the contributions from the least-squares term and the discontinuous Galerkin term, respectively. In the presence of small diffusion (the first term on the right-hand side of (111)) the stability comes primarily from the least-squares and the discontinuous Galerkin terms.

6 Numerical Scheme for Poisson equation

A standard Galerkin finite element formulation is implemented for the Poisson equation. Advanced numerical methods like the Galerkin/least-squares formulation are not needed for the Poisson equation as the Galerkin finite element method is known to be stable for equations of type (4). The Galerkin finite element formulation for the Poisson equation can be summarized as follows:

Let the variational functional spaces S_p (subscript p denotes Poisson equation) and \mathcal{V}_p both consist of continuous functions with square integrable first derivatives. The solution space S_p is the set of all such functions satisfying the essential boundary conditions. The weighting function space \mathcal{V}_p is made up of functions whose value is zero where essential boundary conditions are specified i.e.

$$\{ S_p = \psi \mid \psi \in H^1(\Omega), \psi = g_p \text{ on } \Gamma_g \} \tag{113}$$

$$\{ \mathcal{V}_p = \bar{\psi} \mid \bar{\psi} \in H^1(\Omega), \bar{\psi} = 0 \text{ on } \Gamma_g \} \tag{114}$$

where g_p are the prescribed essential boundary conditions applied on the boundary Γ_g . Consider the following notation for the definition of the weak and Galerkin forms

$$a_p(u, v)_\Omega = \int_\Omega u_{,i} \theta v_{,i} d\Omega \quad (115)$$

$$(u, v)_{\Gamma_{h_i}} = \int_{\Gamma_{h_i}} u v d\Gamma \quad (116)$$

Weak form

The weak form is stated as follows: Given θ , f and h_i , find $\psi \in S_p$ such that for all $\bar{\psi} \in \mathcal{V}_p$

$$B_p(\bar{\psi}, \psi) = L_p(\bar{\psi}) \quad (117)$$

where

$$B_p(\bar{\psi}, \psi) = a_p(\bar{\psi}, \psi)_\Omega \quad (118)$$

$$L_p(\bar{\psi}) = (\bar{\psi}, f)_\Omega + (\bar{\psi}, \theta h_i)_{\Gamma_{h_i}} \quad (119)$$

Note that $h_i = \psi_{,i} n_i$ are the natural boundary conditions prescribed on boundary Γ_{h_i} , and $f = -\varepsilon(c_1 - c_2 - N_D^+ + N_A^-)$.

Galerkin form

Let S_p^h and \mathcal{V}_p^h be the finite-dimensional approximations to S_p and \mathcal{V}_p , respectively. The Galerkin formulation can be stated as follows: Given θ , f and h_i find $\psi^h \in S_p^h$ such that for all $\bar{\psi}^h \in \mathcal{V}_p^h$

$$B_p(\bar{\psi}^h, \psi^h) = L_p(\bar{\psi}^h) \quad (120)$$

Using a standard finite element discretization [14], a matrix form is obtained which is solved for the electrostatic potential ψ^h at all finite element nodes. The electric fields are computed at the center of each element and then projected onto the mesh nodes using smoothing procedures of a least-squares type [20].

6.1 Consistency

The consistency of Equation (117) with the strong form of the boundary value problem may be verified as follows:

$$\begin{aligned}
 B_p(\bar{\psi}, \psi) - L_p(\psi) &= 0 \\
 \Rightarrow a_p(\bar{\psi}, \psi) - (\bar{\psi}, f)_\Omega - (\bar{\psi}, \theta h_i)_{\Gamma_{h_i}} &= 0 \\
 \Rightarrow (\nabla \bar{\psi}, \theta \nabla \psi) - (\bar{\psi}, f)_\Omega - (\bar{\psi}, \theta h_i)_{\Gamma_{h_i}} &= 0 \\
 \Rightarrow (\bar{\psi}, \theta \nabla \psi)_{\Gamma_{h_i}} - (\bar{\psi}, \nabla(\theta \nabla \psi))_\Omega - (\bar{\psi}, f)_\Omega - (\bar{\psi}, \theta h_i)_{\Gamma_{h_i}} &= 0 \\
 \Rightarrow -(\bar{\psi}, (\nabla(\theta \nabla \psi) + f))_\Omega + (\bar{\psi}, \theta(\nabla \psi - h_i))_{\Gamma_{h_i}} &= 0
 \end{aligned} \tag{121}$$

The above equation gives

$$\nabla(\theta \nabla \psi) + f = 0 \quad \text{on } \Omega \tag{122}$$

which is the original equation (Equation (4)) to be solved and

$$\nabla \psi \cdot n_i = h_i \quad \text{on } \Gamma_{h_i} \tag{123}$$

which are the prescribed natural boundary conditions. Hence the consistency to the original form of the equation to be solved is verified.

6.2 Stability

Stability is established as follows:

$$B_p(\bar{\psi}, \bar{\psi}) = a_p(\bar{\psi}, \bar{\psi}) = \theta \|\nabla \bar{\psi}\|_\Omega^2 \tag{124}$$

Equation (124) means that the left-hand side matrix operator is positive definite, which is basically the stability statement for the Galerkin finite element formulation [16].

7 Solution Schemes

The coupled Poisson and the two-carrier hydrodynamic equations are solved employing a staggered scheme, which resembles the popular Gummel procedure [10]. The Poisson equation is first solved for the electrostatic potential. The electric fields are computed from the obtained potential by using smoothing procedures of a least-squares type. The computed electric field values are then used to solve the electron hydrodynamic equations for electron concentration, velocities and temperature. The electron hydrodynamic equations also require the hole concentration and since the hole concentration at the current iteration is not available, the value from the previous iterate is used. We next solve the hole hydrodynamic equations for hole concentration, velocities and temperature. Since the hole hydrodynamic equations are coupled to the electron concentration, either the currently computed electron concentration or the one computed in the previous iteration can be used. A faster convergence can be obtained if the currently available electron concentration is used. The computed concentrations for electrons and holes provide a new source term to the Poisson equation. This procedure of alternatively

solving the Poisson and the electron and hole hydrodynamic equations is repeated until all the equations are solved to a desired tolerance.

A number of advantages can be accounted for the proposed staggered scheme. First, it is simple and the storage requirement is much less than treating the coupled system as a whole. Second, the method converges for almost all arbitrary initial guesses. Third, the separation of the two systems allows the use of efficient solvers developed for each system. For instance, a non symmetric equation solver is needed for the electron and hole hydrodynamic systems but only a symmetric solver is needed for the Poisson equation. For some problems, we can also minimize the cost by solving the Poisson equation as accurate as possible but relaxing the tolerance for the solution of the hydrodynamic system during the iterative process. The fourth advantage is that we can study the error estimators for each system separately, thus simplifying the complexity of the problem. However, by solving all the equations as a single system, a much faster convergence to steady-state solution can be obtained, if good initial guesses can be provided.

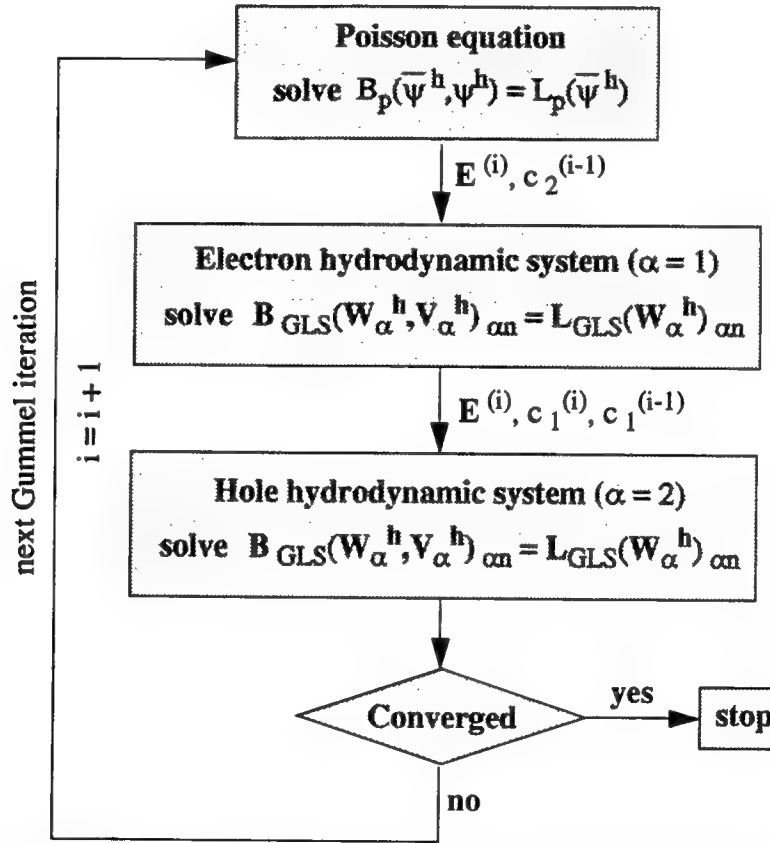


Figure 1 A staggered scheme for solving coupled two-carrier hydrodynamic device equations

8 Numerical Examples

In this section, numerical examples are presented to demonstrate the performance of the finite

element method described in the previous sections. Examples of one and two dimensional two-carrier pn diodes are solved in forward bias. The operation of pn diodes poses several challenges to the numerical schemes as the examples involve localized regions (also termed as depletion regions) where the carrier concentrations vary over several orders of magnitude within a distance of few tenths of a micron. By presenting numerical results for these examples a number of issues are demonstrated: First, the single-carrier formulation can be extended to two-carriers in a straight forward manner. Second the performance of the numerical scheme (stability, robustness and accuracy) are not effected by the addition of a second carrier or the device operation in high level injection and finally, the proposed formulation requires minimal changes to extend the computer program for single carrier simulation to incorporate the second carrier.

8.1 Example 1

The first example is a one-dimensional silicon n^+p diode which is $1.0 \mu\text{m}$ in length and is operated in forward bias. The n^+ -region is doped with $1.0 \times 10^{18} / \text{cm}^3$ and the p-region is doped with $1.0 \times 10^{16} / \text{cm}^3$. The n^+ region is $0.2 \mu\text{m}$ in length and the doping in the n^+p transition region varies as a Gaussian function with $\sigma = 0.01 \mu\text{m}$. The geometry of the diode is shown in Figure 2.

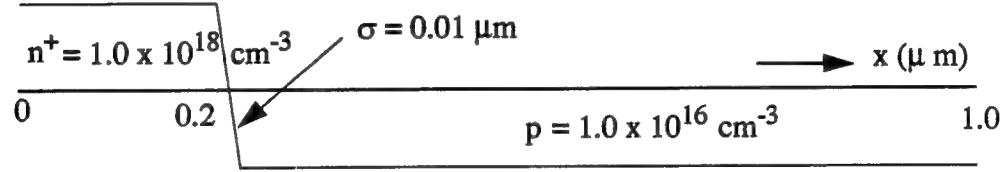


Figure 2: Geometry of an one-dimensional pn diode

The boundary conditions applied are given as follows:

$$\begin{aligned} \text{At } x = 0 \mu\text{m} \quad & c_1 = 9.9 \times 10^{17} \text{ cm}^{-3} \\ & T_1 = T_2 = 300 \text{ K and} \\ & \psi = \psi_b(N_D) \end{aligned}$$

$$\begin{aligned} \text{At } x = 1.0 \mu\text{m} \quad & c_2 = 1.0 \times 10^{16} \text{ cm}^{-3} \\ & T_1 = T_2 = 300 \text{ K and} \\ & \psi = -\psi_b(N_A) + \psi_{appl} \end{aligned}$$

where $\psi_b(N)$ is the built-in potential (a function of doping, N) defined as

$$\psi_b(N) = \frac{k_b T_0}{\epsilon} \ln\left(\frac{N}{c_{int}}\right)$$

and N_D , N_A are the net donor and acceptor concentrations. ψ_{appl} denotes the applied bias which is taken as 1.0 V. The initial conditions for the time-marching scheme that we employ to reach steady state are as follows:

$$\begin{aligned} \text{At } t = 0 \quad c_1(x, 0) &= \frac{N_D - N_A}{2} + \left[\left(\frac{N_D - N_A}{2} \right)^2 + c_{int}^2 \right]^{0.5}; \quad c_2(x, 0) = \frac{c_{int}^2}{c_1(x, 0)} \\ u_1(x, 0) &= u_2(x, 0) = 0.0; \text{ and} \\ T_1(x, 0) &= T_2(x, 0) = T_0 \end{aligned}$$

In this problem a continuation method is used i.e. a bias increment of 0.2V is applied starting at 0 V. We used 1001 mesh points with a uniform mesh spacing of 10 Å. A non uniform mesh of 250 mesh points with finer spacing in the depletion region can also be used to obtain the same accuracy results as shown in this paper. We are currently designing adaptive algorithms to further investigate the issue of optimal meshes without affecting the accuracy of the solution. The steady state results for this problem are shown in Figures 3-10.

Figures 3 and 4 show the electron and hole concentrations, respectively. The electron and hole concentrations vary by several orders of magnitude in a very small localized region. The electron concentration in the p-region and the hole concentration in the n-region increase significantly as the applied bias is increased. Figures 5 and 6 show the electron and hole velocity, respectively. As the applied bias increases the electron velocity increases sharply and steeply near $X = 1 \mu\text{m}$. Also notice the steep drop in the hole velocity near $X = 1 \mu\text{m}$. These velocity components contribute to a significant increase in current as the applied bias is increased. Figures 7 and 8 show the electron and hole temperature, respectively. The electron and hole temperatures undergo rapid changes near $X = 1 \mu\text{m}$ for applied bias of 1.0 volt. This is because of the operation of the diode in high level injection. For low applied biases, small temperature drops can be observed in the depletion region. Figure 9 shows the variation of the electrostatic potential in the diode and Figure 10 shows the variation of the electric field which is the negative gradient of potential.

8.2 Example 2

The second example is a two dimensional silicon pn diode which is $3.5 \mu\text{m} \times 2.5 \mu\text{m}$. The n^+ -region has a doping of $1.0 \times 10^{17} \text{ cm}^{-3}$ and the p-region has a doping of $1.0 \times 10^{15} \text{ cm}^{-3}$. The transition between the n^+ and p region is not abrupt and is treated as a Gaussian variation with $\sigma = 0.4 \mu\text{m}$. Two contacts are placed along the boundaries of the device and the device is operated in forward bias. Both contacts are assumed to be ohmic. The geometry of the diode and the placement of the contacts are shown in Figure 11.

The n contact extends up to a distance of $0.5 \mu\text{m}$ from the top left corner and the p contact covers the entire base. For forward bias operation of the diode, 0.0 V is applied on the n contact and 0.8V is applied on the p contact. The boundary conditions are applied as follows:

- i) Along contact 1-2: $c_2 = 1.0 \times 10^{15} \text{ cm}^{-3}$, $u_2 = 0 \text{ cm/s}$ and $\psi = -\psi_b(N_A) + 0.8V$.
- ii) Along boundaries 2-3 & 1-5: $u_1 = u_2 = 0 \text{ cm/s}$ and $\frac{\partial \psi}{\partial n} = 0$ (Neumann boundary condition for potential).

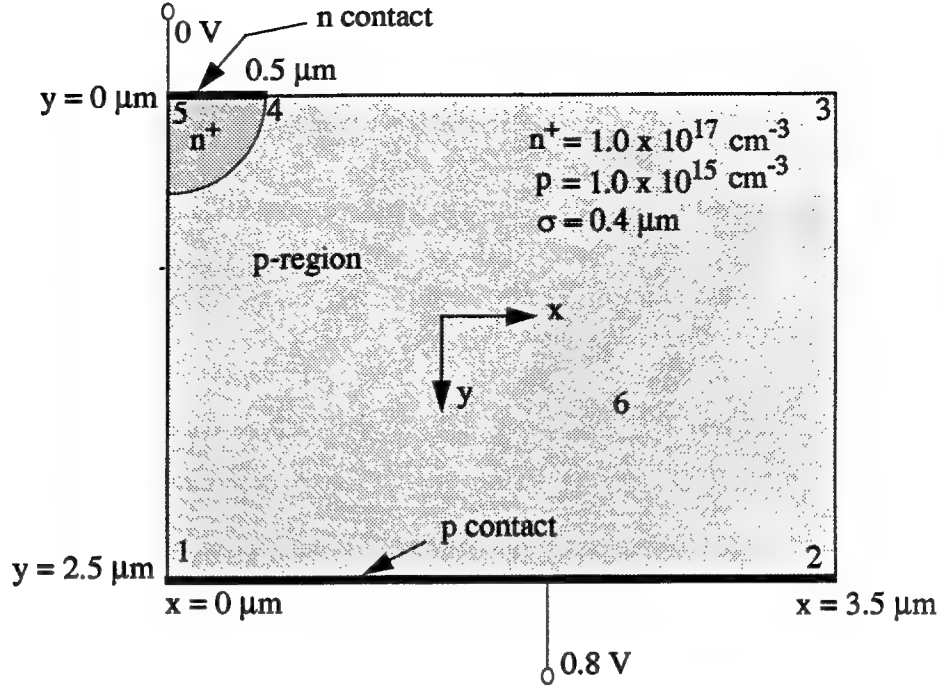


Figure 11: Geometry of a two-dimensional pn diode

iii) Along boundary 3-4: $v_1 = v_2 = 0$ cm/s and $\frac{\partial \psi}{\partial n} = 0$.

iv) Along contact 4-5: $c_1 = 1.0 \times 10^{17}$ cm⁻³, $T_1 = T_0$ and $\psi = \psi_b(N_D)$.

Note that the boundary conditions specified above do not necessarily follow the boundary conditions discussed in Section 4.4. This is because 0.8V can be considered as a high forward bias or a high level injection case and in this case $c_1 c_2 = c_{int}^2$ is not a reasonable approximation and hence the concentration for minority carriers is typically not known at outflow boundaries. Hence, in order to strictly impose the correct number of boundary conditions, mixed type of boundary conditions or their variants are needed and this can be quite challenging. Instead, we have imposed the boundary conditions in terms of the quantities that are generally known. This method of specification can lead to over specified or under specified systems of equations and robust numerical schemes are needed to guarantee convergence. Our results indicate that the numerical scheme proposed in this paper is not very sensitive to the specification of boundary conditions. However, the convergence of the algorithm could be slow. Adhering to the strict imposition of boundary conditions discussed in Section 4.4 can lead to boundary layers near the contacts. The reader should however note that the boundary conditions discussed in Section 4.4 are applied to low forward bias regime.

The initial conditions are given as follows:

$$\begin{aligned} \text{At } t = 0 \quad c_1(x, y, 0) &= \frac{N_D - N_A}{2} + \left[\left(\frac{N_D - N_A}{2} \right)^2 + c_{int}^2 \right]^{0.5}, \quad c_2(x, y, 0) = \frac{c_{int}^2}{c_1(x, y, 0)} \\ u_1(x, y, 0) &= u_2(x, y, 0) = 0.0 \\ v_1(x, y, 0) &= v_2(x, y, 0) = 0.0 \end{aligned}$$

$$T_1(x, y, 0) = T_2(x, y, 0) = T_0$$

For this problem a continuation method is used with a bias increment of 0.1V starting from 0V. A mesh of 64 x 47 nodes is employed. The steady state results for this problem are shown in Figures 12-22.

Figures 12 and 13 show the electron and hole concentrations, respectively. Similar to the one dimensional example the electron and hole concentrations vary over several orders of magnitude in small localized regions and the numerical algorithm proposed is able to resolve such a sharp gradient effectively. Figures 14 and 15 show the electron and hole velocities in the x-direction, respectively. Velocity overshoot can be observed close to the termination of n-contact. The velocity overshoot could be the result of the discontinuity in the velocity boundary condition. This velocity overshoot phenomenon does not occur in low forward bias cases. Figures 16 and 17 show the electron and hole velocities in the y-direction, respectively. The electron and hole temperatures shown in Figures 18 and 19, respectively, indicate that the electrons get heated more than the holes. The hole temperatures are very close to the room temperatures while the electron temperatures are slightly higher in the p-region. Figure 20 shows the electrostatic potential and Figures 21 and 22 show the electric fields in the x and y directions, respectively.

9 Conclusions

A space-time Galerkin/least-squares finite element method, proposed and implemented for two-carrier hydrodynamic equations, is able to solve the coupled semiconductor device equations efficiently and accurately. The proposed numerical algorithms are shown to be stable and consistent. A Clausius-Duhem inequality is derived for the hydrodynamic conservation laws and the entropy variable based approach is shown to automatically satisfy this inequality.

Theoretical results for boundary conditions are derived for the well-posedness of the hydrodynamic model. The practical difficulty in imposing the theoretically observed results is addressed for high forward bias voltages. A bridge needs to be built between theory and practice for special cases and this is a topic for further investigation.

In earlier papers [1], [8], [9] it was observed that the heat conduction term plays a very important role that can significantly affect the accuracy of the solution. Hence, new models have been proposed in which the coefficient of heat conductivity is reduced [11]. For the numerical examples shown in this paper, it was observed that the results are not significantly different with old and new heat conduction models.

The numerical scheme proposed in this paper is computationally very intensive. Several hours of computing time could be needed if the simulations were to be performed on workstations. Parallel algorithms have been developed and implemented to efficiently solve complex device examples on state-of-the art parallel machines. A discussion of the parallel implementation on a MIMD distributed memory computer is beyond the scope of this paper. Our current and future efforts involve the design and development of adaptive, parallel adaptive algorithms and three dimensional device simulation.

10 Acknowledgments

The authors would like to thank Prof. T. J. R. Hughes for helpful discussions on the finite element formulation for two-carrier devices and Drs. Ke-Chih Wu, Zhiping Yu and Edwin Kan for many helpful suggestions. This research is sponsored by ARPA through contract #DAAL 03-91-C-0043.

References

- [1] N. R. Aluru, A. Raefsky, P. M. Pinsky, K. H. Law, R. J. G. Goossens and R. W. Dutton, "A finite element formulation for the hydrodynamic semiconductor device equations", *Comp. Meth. Appl. Mech. Engg.*, vol. 107, pp. 269-298, 1993.
- [2] N. R. Aluru, K. H. Law, P. M. Pinsky, A. Raefsky, R. J. G. Goossens and R. W. Dutton, "Space-Time Galerkin/Least-Squares finite element formulation for the hydrodynamic device equations", *IEICE Trans.-Electron.*, vol. E77-C, No. 2, pp. 227-235, 1994.
- [3] N. R. Aluru and K. H. Law, "A study on the well-posed boundary conditions for the full hydrodynamic model of semiconductor devices", Technical Report - October 1993, Integrated Circuits Laboratory, Stanford University, Stanford, CA
- [4] G. Baccarani and M. R. Wordeman, "An investigation of steady-state velocity overshoot in silicon", *Solid-State Electronics*, vol. 28, pp. 407-416, 1985.
- [5] K. Blotekjaer, "Transport equations for electrons in two-valley semiconductors", *IEEE Transactions on Elec. Dev.*, vol. ED-17, pp. 38-47, 1970.
- [6] S. W. Bova and G. F. Carey, "An analysis of the time-dependent hydrodynamic device equations", *Proc. Intl. Workshop Comp. Elec.*, Univ. of Illinois, pp. 91-94, May 28-29, 1992.
- [7] D. Chen, E. C. Kan, U. Ravaioli, Z. Yu and R. W. Dutton, "A self-consistent discretization scheme for current and energy transport equations", presented at IV Int. Conf. on Simulation of Semiconductor Devices and Processes (SISDEP), Zurich, Sep. 12-14, 1991.
- [8] E. Fatemi, J. W. Jerome and S. Osher, "Solution of the hydrodynamic device model using high-order nonoscillatory shock capturing algorithms", *IEEE Transactions on CAD*, vol. 10, pp. 232-244, 1991.
- [9] C. L. Gardner, J. W. Jerome and D. J. Rose, "Numerical methods for the hydrodynamic device model: subsonic flow", *IEEE Transactions on CAD*, vol. 8, pp. 501-507, 1989.
- [10] K. K. Gummel, "A self-consistent iterative scheme for one-dimensional steady state transistor calculations", *IEEE Trans. Elec. Dev.*, vol. ED-11, pp. 455-465, 1964.
- [11] A. Gnudi, F. Odeh and M. Rudan, "Investigation of non-local transport phenomena in small semiconductor devices", *European Trans. Telecomm.*, vol. 1, 307-313, 1990.
- [12] B. Gustafson and A. Sundstrom, "Incompletely parabolic problems in fluid dynamics", *SIAM J. Appl. Math.*, vol. 35, pp. 343-357, 1978.
- [13] A. Harten, "On the symmetric form of the systems of conservation laws with entropy", *J. Comp. Phys.*, vol. 49, pp. 151-164, 1983.
- [14] T. J. R. Hughes, "The finite element method: Linear static and dynamic finite element analysis", Prentice Hall, Englewood Cliffs, NJ, 1987.
- [15] T. J. R. Hughes and A. Brooks, "A theoretical framework for Petrov-Galerkin methods with discontinuous weighting functions. Application to the streamline upwind procedure", in Gallagher et al. eds., *Finite Elements in Fluids*, vol. 4, pp. 47-65, 1982.
- [16] T. J. R. Hughes, L. P. Franca and G. M. Hulbert, "A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive sys-

- tems", *Comp. Meth. Appl. Mech. Engg.*, vol. 58, pp. 173-189, 1986.
- [17] T. J. R. Hughes, L. P. Franca and M. Mallet, "A new finite element formulation for computational fluid dynamics: I. Symmetric forms of the compressible and Euler and Navier-Stokes equations and the second law of thermodynamics", *Comp. Meth. Appl. Mech. Engg.*, vol. 54, pp. 223-234, 1986.
 - [18] T. J. R. Hughes and M. Mallet, "A new finite element formulation for computational fluid dynamics: III. The generalized streamline operator for multidimensional advective-diffusive systems", *Comp. Meth. Appl. Mech. Engg.*, vol. 58, pp. 305-328, 1986.
 - [19] C. Johnson, U. Navert and J. Pitkaranta, "Finite element methods for linear hyperbolic problems", *Comp. Meth. Appl. Mech. Engg.*, vol. 45, pp. 285-312, 1984.
 - [20] R. L. Lee, P. M. Gresho and R. L. Sani, "Smoothing techniques for certain primitive variable solutions of the Navier-Stokes equations", *Intl. J. for Num. Meth. Engg.*, vol. 14, pp. 1785-1804, 1979.
 - [21] J. E. Marsden and T. J. R. Hughes, "Mathematical foundations of elasticity", Prentice-Hall, Englewood Cliffs, NJ, 1983.
 - [22] J. Oliger and A. Sundstrom, "Theoretical and practical aspects of some initial boundary value problems in fluid dynamics", *SIAM J. Appl. Math.*, vol. 35, pp. 419-446, 1978.
 - [23] M. R. Pinto, "Comprehensive semiconductor device simulation for silicon ULSI", Dept. of Elec. Engg., Ph.D. Thesis, Stanford University, Aug. 1990.
 - [24] M. Rudan, F. Odeh and J. White, "Numerical solution of the hydrodynamic model for a one-dimensional device", *COMPEL*, vol. 6, pp. 151-170, 1987.
 - [25] S. Selberherr, *An Analysis and simulation of semiconductor devices*. New York: Springer-Verlag, 1984.
 - [26] F. Shakib, "Finite element analysis of the compressible Euler and Navier-Stokes equations", Dept. of Mech. Engg., Ph.D Thesis, Stanford University, Nov. 1988.
 - [27] F. Shakib, T. J. R. Hughes and Z. Johan, "A new finite element formulation for computational fluid dynamics: X. The compressible Euler and Navier-Stokes equations", *Comp. Meth. Appl. Mech. Engg.*, vol. 89, pp. 141-219, 1991.
 - [28] M. Sharma and G. F. Carey, "Semiconductor device modeling using flux upwind finite elements", *COMPEL*, vol. 8, pp. 219-224, 1989.
 - [29] J. Strikwerda, "Initial boundary value problems for incompletely parabolic systems", Ph.D. thesis, Dept. of Math., Stanford Univ., Stanford, CA, 1976.
 - [30] E. Thomann and F. Odeh, "On the well-posedness of the two-dimensional hydrodynamic model for semiconductor devices", *COMPEL*, vol. 9, pp. 45-57, 1990.

Appendix

Lemma: $a_{\alpha n} = \int_{\Omega} V^h(t_n^+) [[U(V^h(t_n))]] d\Omega - \int_{\Omega} (\mathcal{H}^h(t_n^+) - \mathcal{H}^h(t_n^-)) d\Omega \geq 0$

Proof:

Using Taylor's formula with integral form of remainder

$$\begin{aligned} \mathcal{H}^h(t_n^-) - \mathcal{H}^h(t_n^+) + V^h(t_n^+) [[U(V^h(t_n))]] \\ = \int_0^1 (1-\varepsilon) [[U(V^h(t_n))]] \cdot \tilde{A}_0^{-1}(U(t_n^+) - \varepsilon [[U(t_n)])) [[U(t_n)]] d\varepsilon \\ \geq \hat{c} |[[U(t_n)]]|_{\tilde{A}_0^{-1}}^2 \end{aligned}$$

where $|X|_{\tilde{A}_0^{-1}}^2 = X \cdot \tilde{A}_0^{-1} X$

Therefore, $a_{\alpha n} = \int_{\Omega} V^h(t_n^+) [[U(V^h(t_n))]] d\Omega - \int_{\Omega} (\mathcal{H}^h(t_n^+) - \mathcal{H}^h(t_n^-)) d\Omega \geq 0$

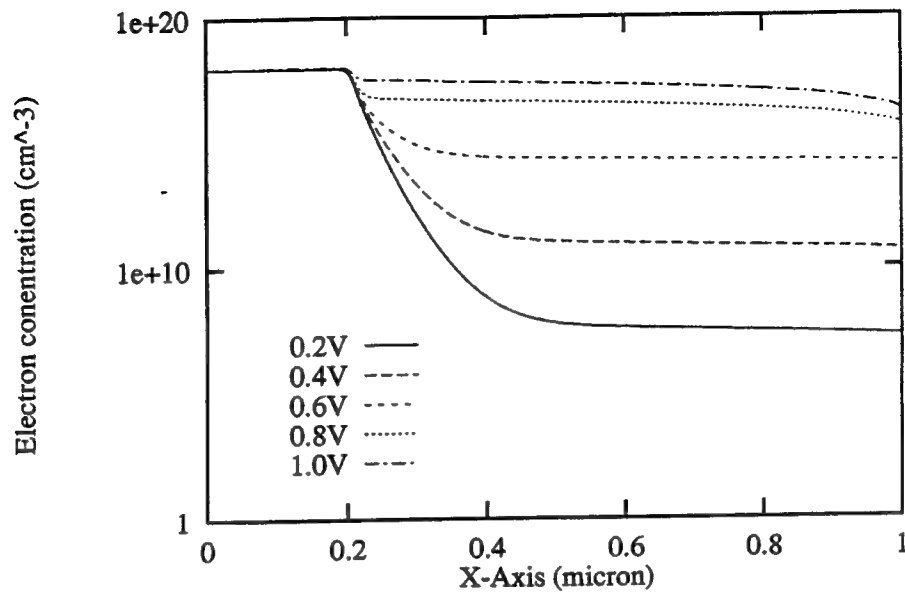


Figure 3: Electron concentration (cm^{-3}) in steady state for forward biases of 0.2V to 1.0V with 0.2V increments

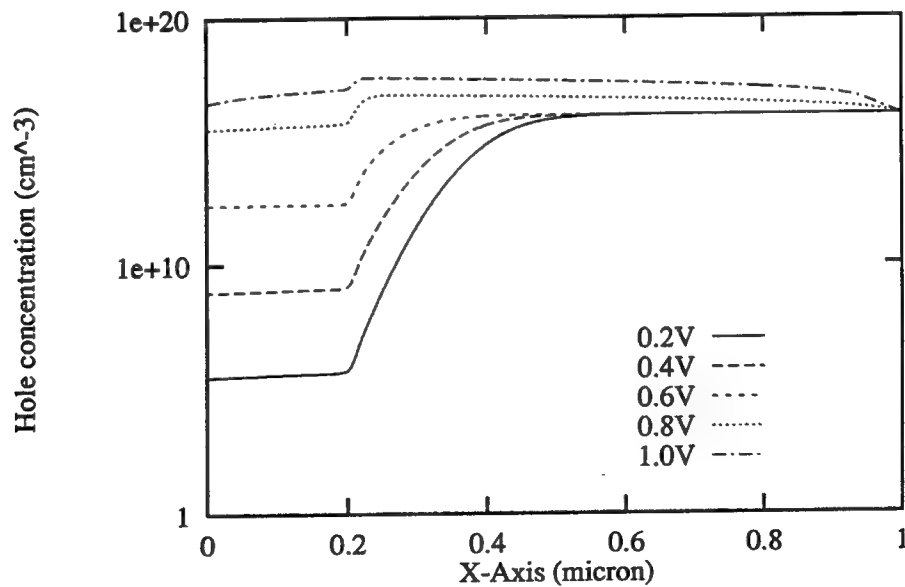


Figure 4: Hole concentration (cm^{-3}) in steady state for forward biases of 0.2V to 1.0V with 0.2V increments

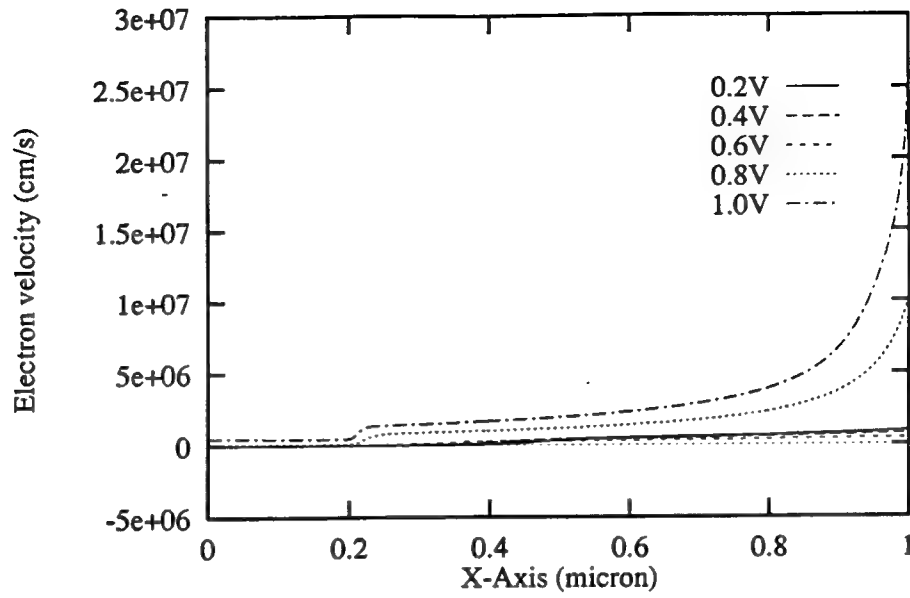


Figure 5: Electron velocity (cm/s) in steady state for forward biases of 0.2 to 1.0V with 0.2V increments

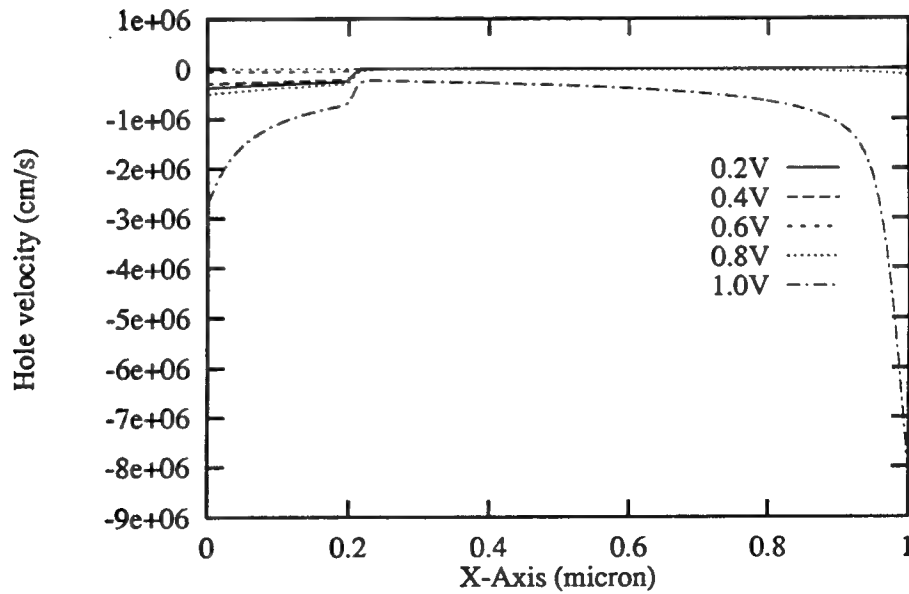


Figure 6: Hole velocity (cm/s) in steady state for forward biases of 0.2 to 1.0V with 0.2V increments

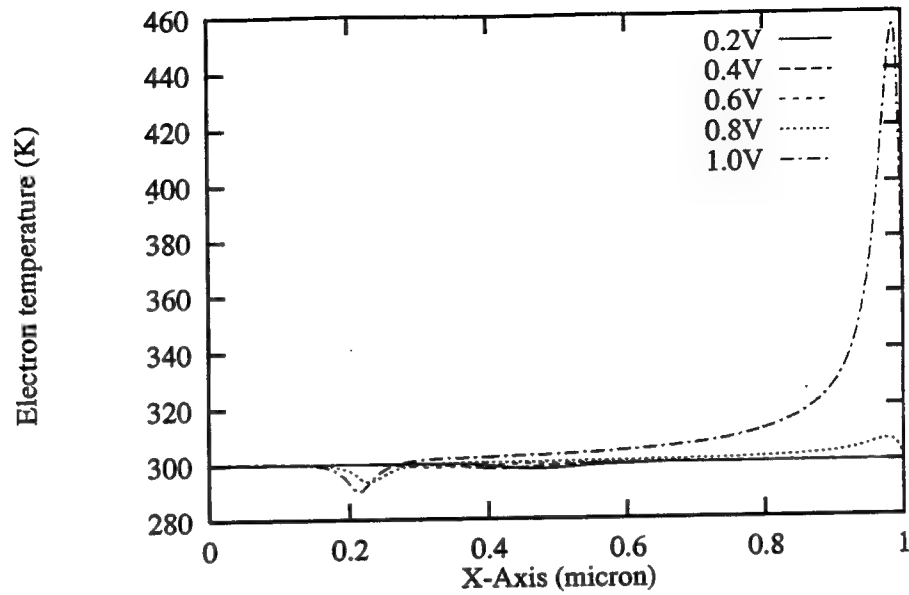


Figure 7: Electron temperature (K) in steady state for forward biases of 0.2V to 1.0V with 0.2V increments

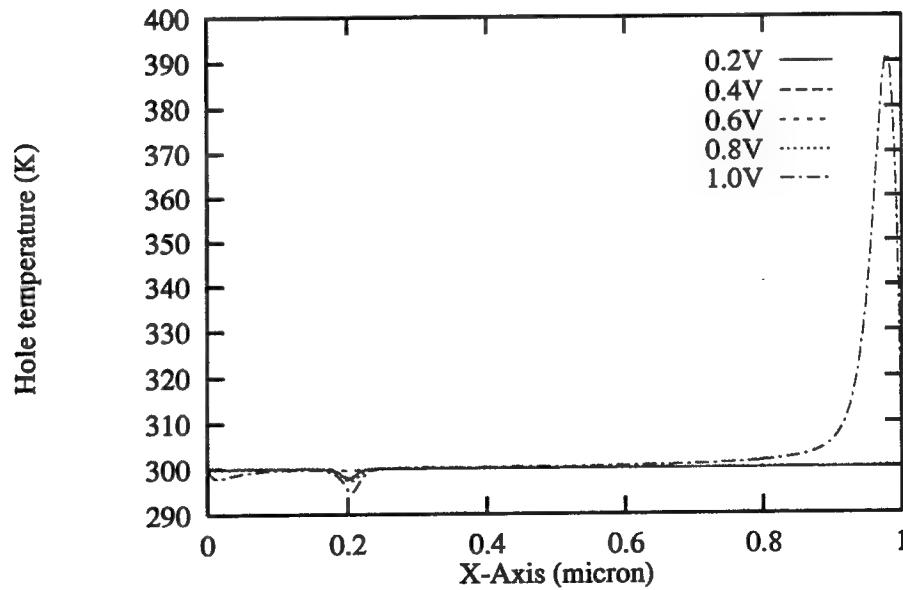


Figure 8: Hole temperature (K) in steady state for forward biases of 0.2V to 1.0V with 0.2V increments

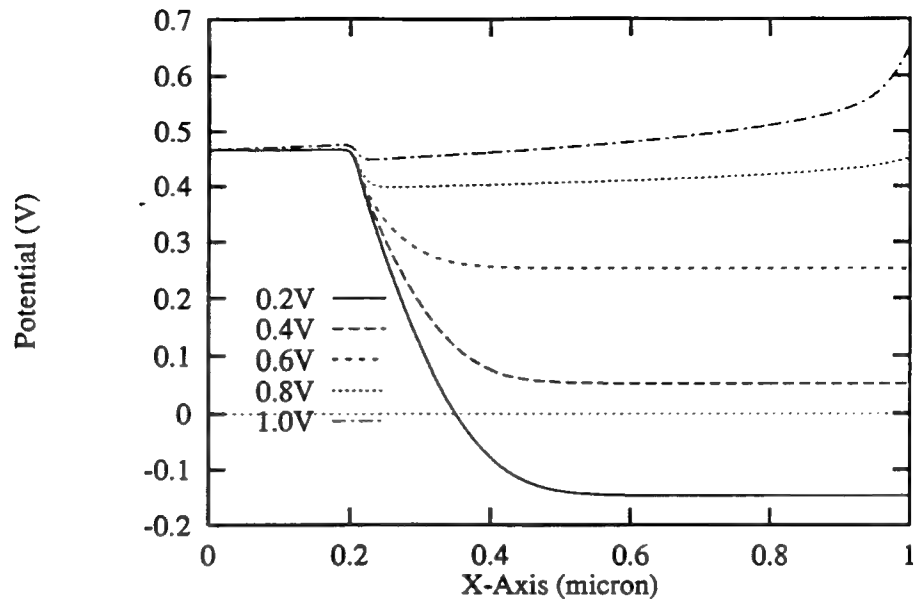


Figure 9: Electrostatic potential (V) in steady state for forward biases of 0.2V to 1.0V with 0.2V increments

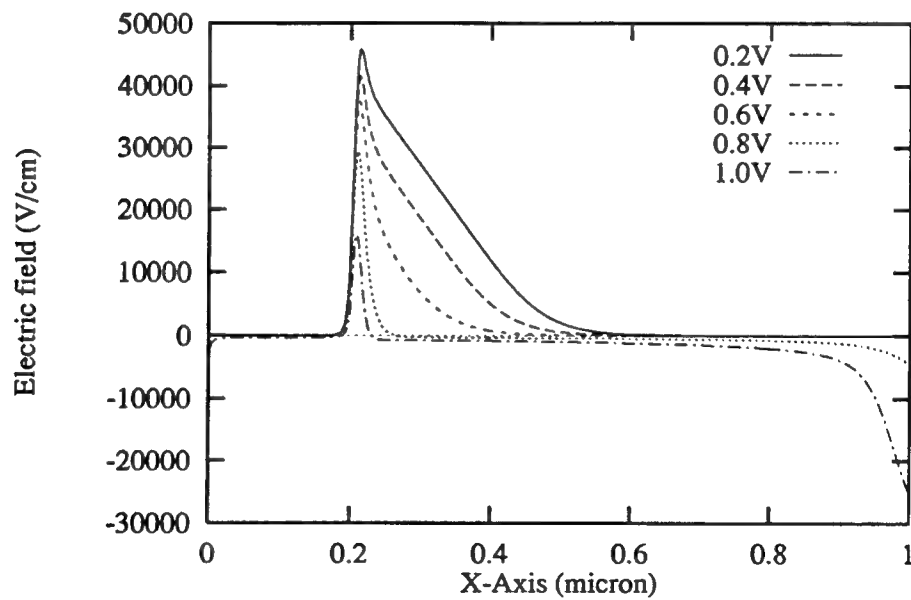


Figure 10: Electric field (V/cm) in steady state for forward biases of 0.2V to 1.0V with 0.2V increments

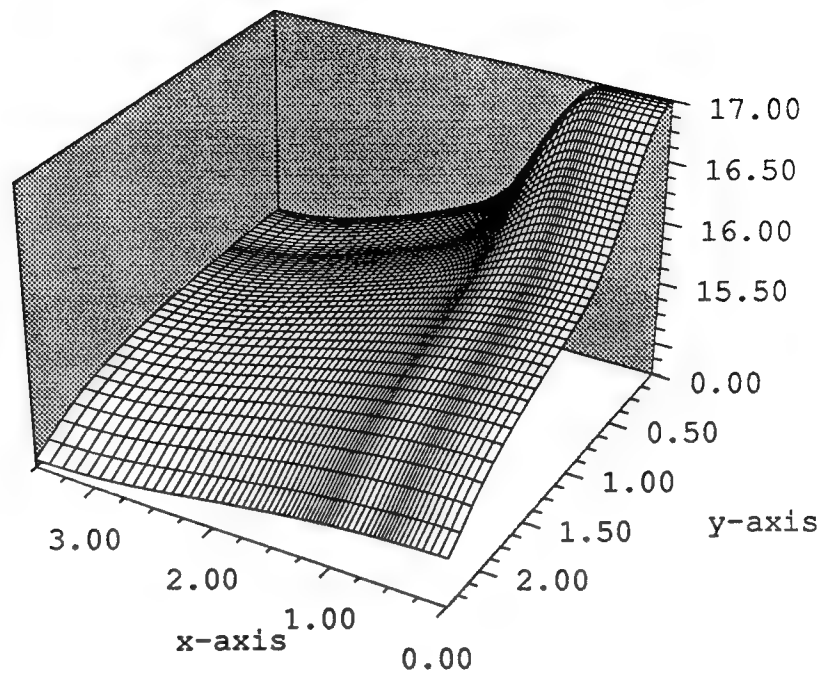


Figure 12: Electron concentration (cm^{-3}) in steady state for 2D pn diode in forward bias of 0.8V.

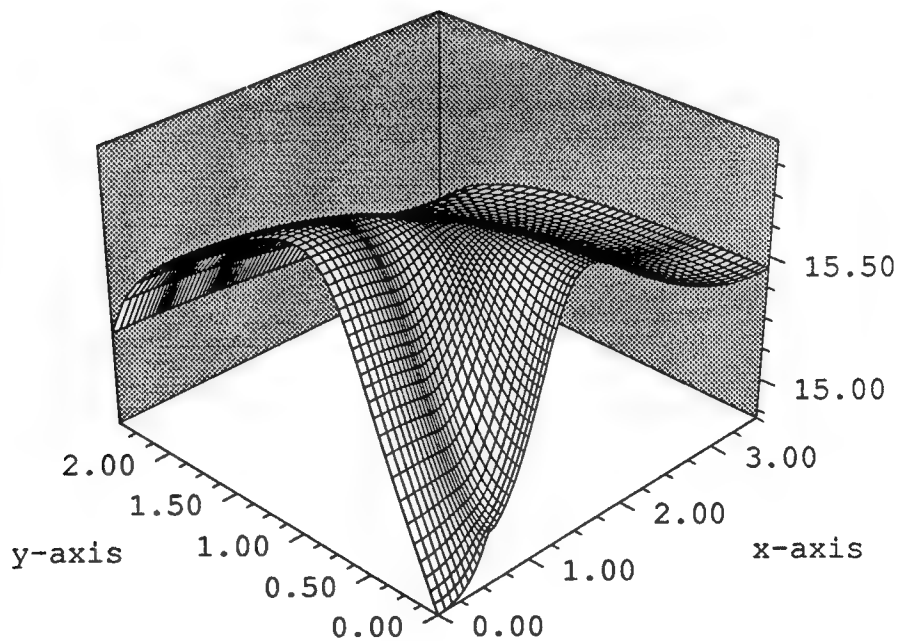


Figure 13: Hole concentration (cm^{-3}) in steady state for 2D pn diode in forward bias of 0.8V.

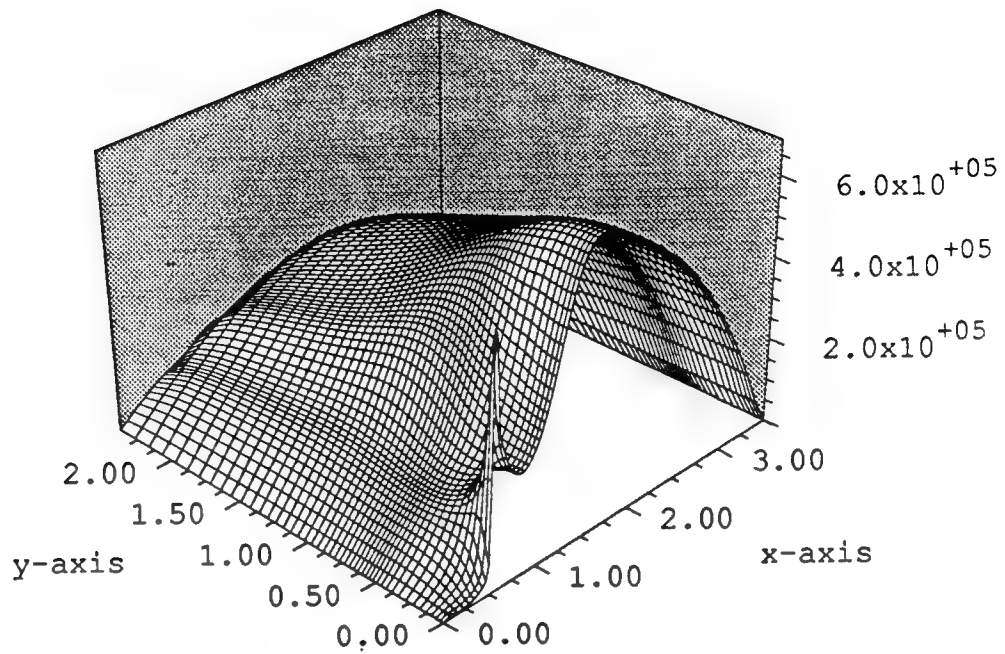


Figure 14: Electron x-velocity (cm/s) in steady state for 2D pn diode in forward biases of 0.8V.

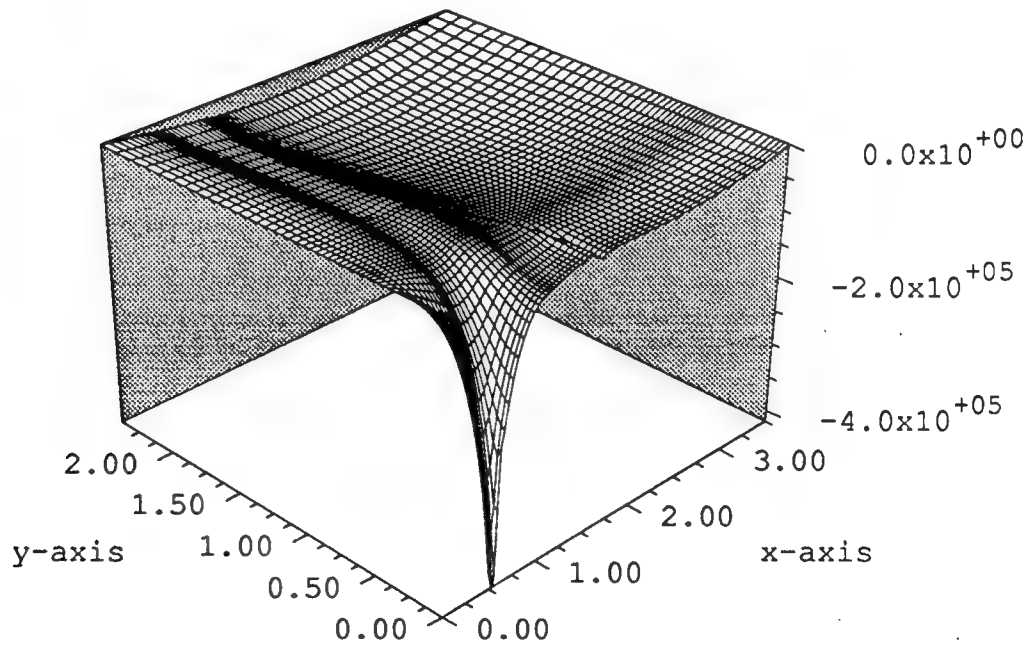


Figure 15: Hole x-velocity (cm/s) in steady state for 2D pn diode in forward biases of 0.8V.

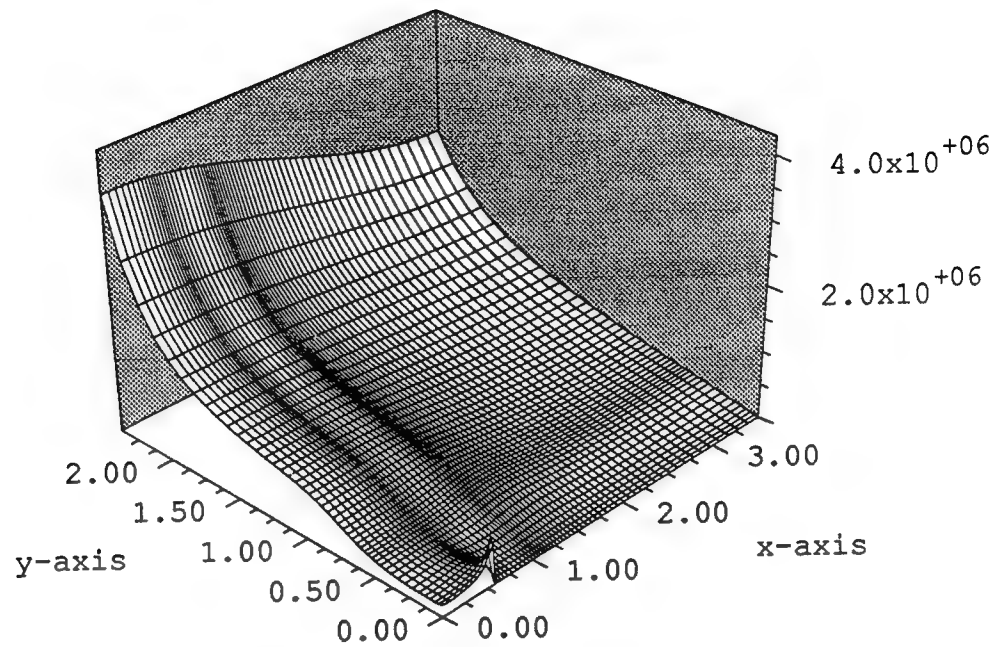


Figure 16: Electron y-velocity (cm/s) in steady state for 2D pn diode in forward biases of 0.8V

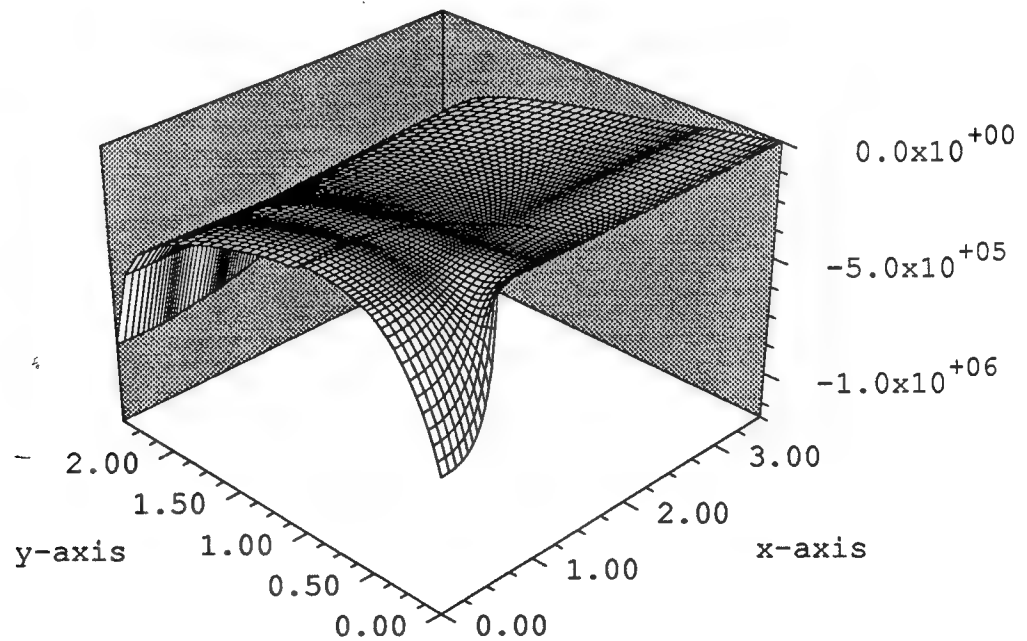


Figure 17: Hole y-velocity (cm/s) in steady state for 2D pn diode in forward biases of 0.8V

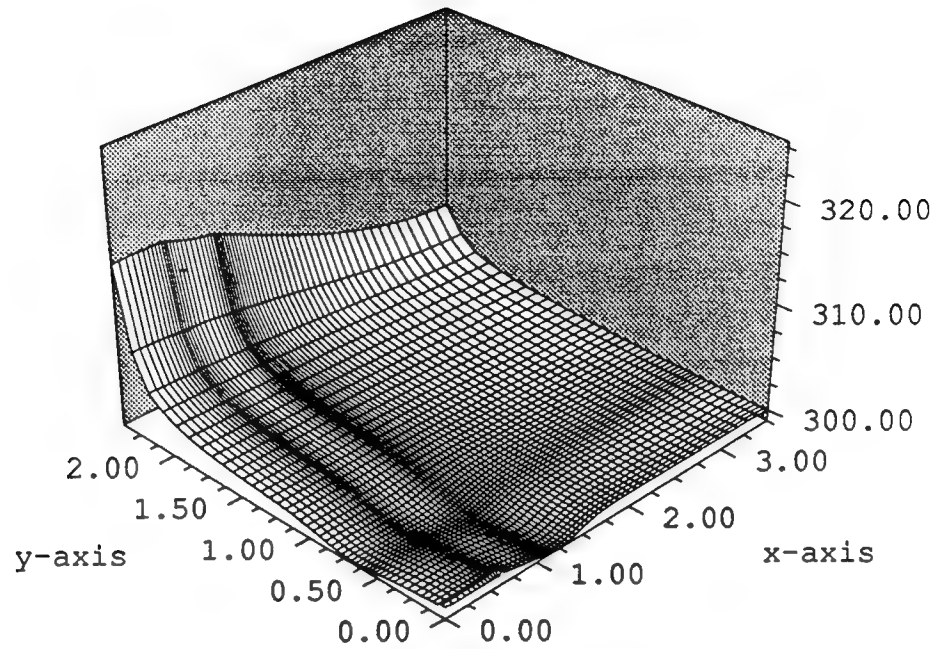


Figure 18: Electron temperature (K) in steady state for 2D pn diode in forward biases of 0.8V.

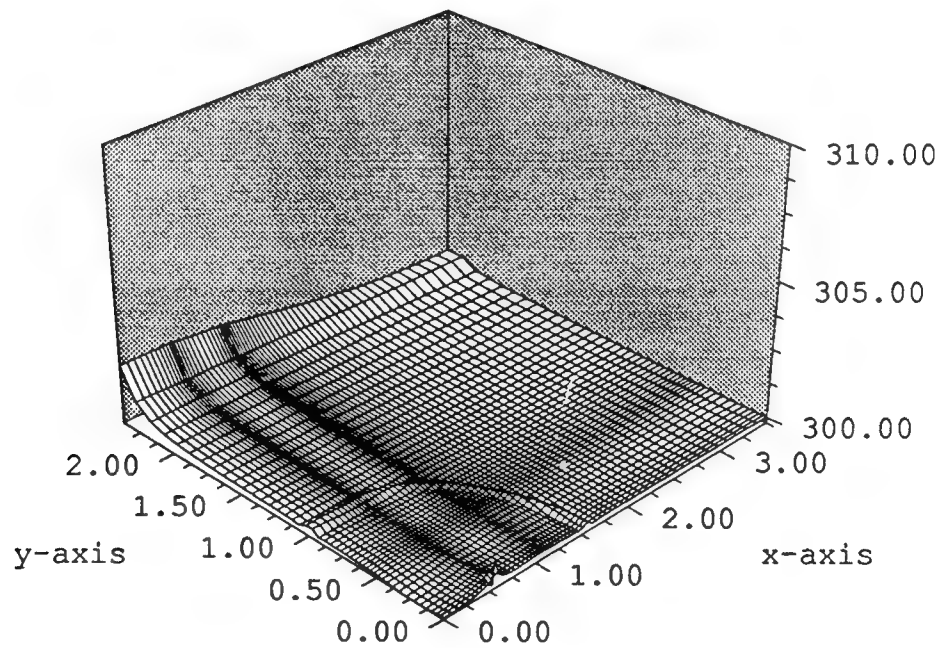


Figure 19: Hole temperature (K) in steady state for 2D pn diode in forward biases of 0.8V.

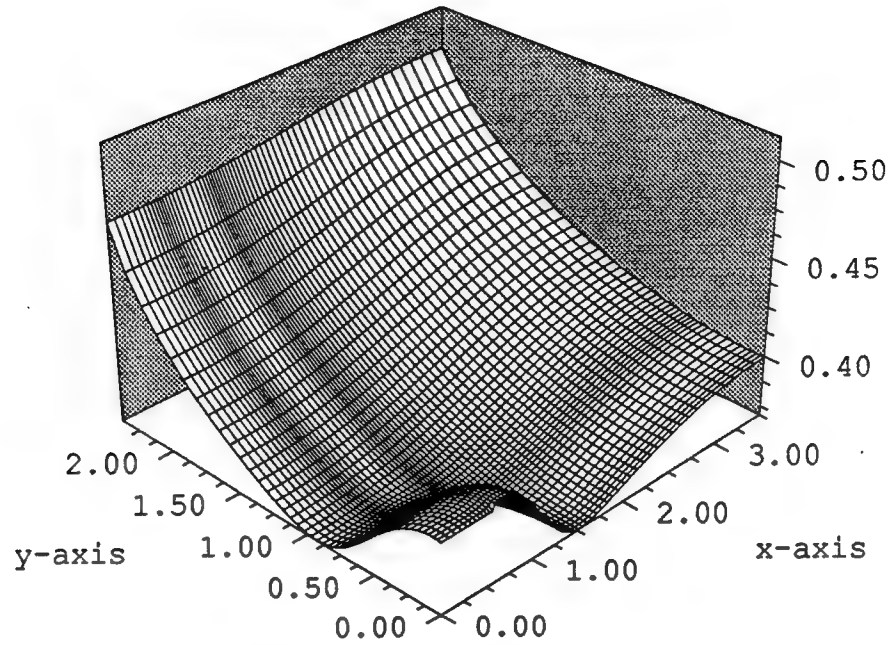


Figure 20: Electrostatic potential (V) in steady state for 2D pn diode in forward biases of 0.8V.

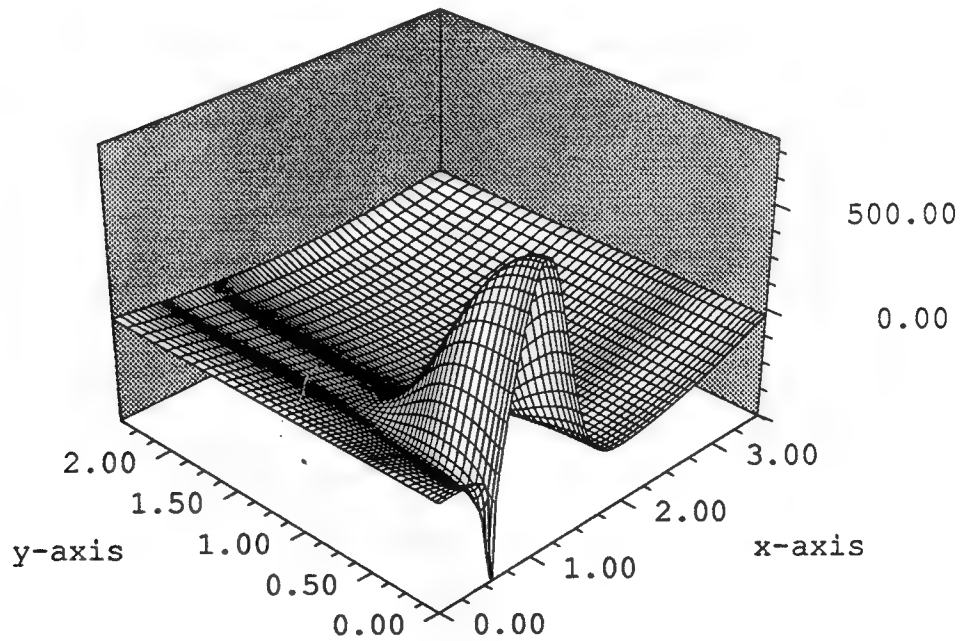


Figure 21: X - Electric field (V/cm) in steady state for 2D pn diode in forward biases of 0.8V.

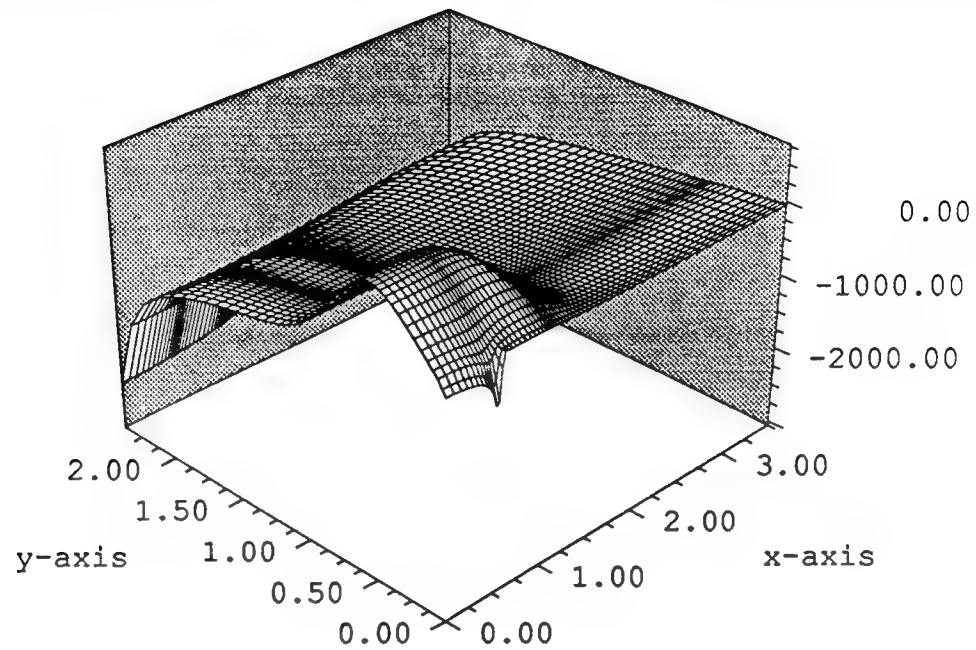


Figure 22: Y - Electric field (V/cm) in steady state for 2D pn diode in forward biases of 0.8V.

An Analysis of the Hydrodynamic Semiconductor Device Model - Boundary Conditions and Simulations

N. R. Aluru, K. H. Law, P. M. Pinsky and R. W. Dutton
Integrated Circuits Laboratory
Stanford University, Stanford, California, USA 94305

October, 1994.

Abstract

A mathematical analysis of the time-dependent multi-dimensional Hydrodynamic model is performed to determine the well-posed boundary conditions for semiconductor device simulation. The number of independent boundary conditions that need to be specified at electrical contacts of a semiconductor device are derived. Using the classical energy method, a mathematical relation among the physical parameters is established to define the well-posed boundary conditions for the problem. Several possible sets of boundary conditions are given to illustrate the proper boundary conditions. Natural boundary conditions that can be specified are obtained from the boundary integrals of the weak-form finite element formulations. An example is included to illustrate the importance of well-posedness of the boundary conditions for device simulation.

1 Introduction

Semiconductor device simulation has been based primarily on the drift-diffusion (DD) model for carrier transport, a simplification of the Boltzmann Transport Equation (BTE). With the scaling of silicon devices into deep submicron region, non-stationary phenomena such as velocity overshoot and carrier heating are becoming increasingly important to determine the characteristics of these devices. Due to the assumption of local equilibrium, the DD model cannot capture such non-stationary phenomena accurately. Although the direct solution of BTE, for example via Monte Carlo method, can capture the above phenomena, the noise in the solution and the computational cost prevent it from wide usage for device simulation. An attractive alternative is to employ full Hydrodynamic (HD) [1] or HD-like models. The full HD model can be directly derived from the zero, first and second moments of the BTE with a few simplifying assumptions [2]. These equations have a direct analogy to fluid dynamic equations. In this paper we discuss the mathematical development on the well-posedness of the HD model.

Boundaries encountered in semiconductor devices can be classified into two types: The first are the physical boundaries such as electrical contacts and interfaces to insulating material; the second are the artificial boundaries which are introduced to separate neighboring devices in integrated circuits. Well-posed boundary conditions for contacts play an important role in numerical simulations. Prescribing too many boundary conditions precludes the existence of smooth solutions and specifying too few boundary conditions, on the other hand, precludes uniqueness of the solution. More importantly, improper number of boundary conditions dramatically affects the convergence of the numerical schemes. Hence, it is important that the proper set of boundary conditions be specified for numerical simulations.

Well-posed boundary conditions for the classical DD model are well understood. The same set of boundary conditions, however, do not give well-posedness for the HD model. Thomann and Odeh [3] have shown that the boundary conditions based on the DD model are not sufficient for the HD model. While they have shown that additional boundary conditions are needed for the HD model, their analysis has been focused on the 2D hydrodynamic model and for subsonic flows. In [12] Sever presented a study on the well-posedness of the HD model. The boundary conditions suggested in [12] are again valid only for subsonic flows; more importantly, the suggested boundary conditions cannot be implemented easily in the context of semiconductor devices. The issue of the number of boundary conditions that need to be specified at contacts has also not been addressed for multi-dimensional flows in that study. Sever's approach to well-posedness, boundary conditions and discretization is based on symmetrizing the HD equations by employing entropy variables. Unfortunately, the study fell short in application to performing actual device simulations, leading to the question regarding the value of the mathematical results presented for HD equations. It was suggested that the discretized equations obtained using entropy variables are too complex and are impractical for HD equations or for the Euler and Navier-Stokes equations. This implication is clearly unjustified as evident from the work by Hughes and co-workers for Navier-Stokes equations (see [13] and references therein) and for the HD equations [8, 11].

Bova and Carey [4] have reported a study on boundary conditions for HD equations, taking advantage of the resemblance of HD equations to compressible Euler and Navier-Stokes equations.

The number of boundary conditions that they have proposed are identical to those specified for Euler equations. In doing so they assumed that the diffusive effect of the heat flux term on the average energy is small on the boundaries; however, this assumption is lack of physical basis. As shall be shown in this paper, the proper number of boundary conditions that need to be specified for the HD equations are not identical to those of the Euler or Navier-Stokes equations.

Well-posed boundary conditions for Euler and Navier-Stokes equations have been investigated by Strikwerda [5], Gustafson and Sundstrom [6], Olinger and Sundstrom [7], among others. We extend the concepts developed in these studies to derive well-posed boundary conditions for the HD equations. In this paper we describe a general multi-dimensional (one, two and three dimensional) analysis of the HD equations, to include the heat flux term and to place no restriction on the type of flow, albeit subsonic or supersonic nature. A well-posedness condition involving physical parameters is derived and practical difficulties in specifying some sets of boundary conditions that satisfy the well-posedness condition are addressed.

This paper is organized as follows: In section 2 we review the partial differential equations for the hydrodynamic model of semiconductor devices. In section 3 we express the HD equations in terms of a set of primitive variables. In section 4 the number of independent boundary conditions that need to be specified for well-posedness are derived for multi-dimensional HD equations. In section 5 two symmetrization procedures for HD equations are discussed and the energy estimates and inequalities to be satisfied are derived using the classical energy method. A brief discussion is also provided on constructing a finite element formulation and this leads to the discussion on natural boundary conditions. In section 6 examples for various sets of boundary conditions are discussed. Section 7 provides a discussion on natural boundary conditions. In section 8 simulation results are presented for a 0.6 μm MESFET device to illustrate the importance of proper boundary conditions. Finally, we summarize the results of this study in section 9.

2 Device Equations

Semiconductor devices can be simulated by solving the coupled Poisson and HD equations. For single carrier devices, the transport equations for electron gas described by the HD model are summarized as follows:

$$\frac{\partial n}{\partial t} + \nabla \cdot (nu) = \left[\frac{\partial n}{\partial t} \right]_{col} \quad (1)$$

$$\frac{\partial p}{\partial t} + u \cdot (\nabla \cdot p) + (p \cdot \nabla) u = -\epsilon n E - \nabla \cdot (nk_b T) + \left[\frac{\partial p}{\partial t} \right]_{col} \quad (2)$$

$$\frac{\partial w}{\partial t} + \nabla \cdot (uw) = -\epsilon n (u \cdot E) - \nabla \cdot (unk_b T) - \nabla \cdot q + \left[\frac{\partial w}{\partial t} \right]_{col} \quad (3)$$

Equations (1), (2), and (3) are the particle continuity and conservation laws for electron momentum and energy, respectively. In the above equations, n is the concentration of electrons; \mathbf{u} is the electron velocity vector; \mathbf{p} is the electron momentum density vector; T is the electron temperature; w is the electron energy density; \mathbf{q} is the electron heat flux vector; \mathbf{E} is the electric field; ϵ is the magnitude of an elementary charge; k_b is the Boltzmann constant and $[\]_{col}$ denotes collision terms. Equations (1)-(3) represent a system of three partial differential equations with 5 unknowns n , \mathbf{u} , \mathbf{p} , T and w . The following definitions are given for the collision terms appearing in the above equations

$$\left[\frac{\partial n}{\partial t} \right]_{col} = 0 \quad (4)$$

$$\left[\frac{\partial \mathbf{p}}{\partial t} \right]_{col} = \frac{-\mathbf{p}}{\tau_p} \quad (5)$$

$$\left[\frac{\partial w}{\partial t} \right]_{col} = \frac{-\left(w - \frac{3}{2} n k_b T_0 \right)}{\tau_w} \quad (6)$$

where τ_p , τ_w are momentum and energy relaxation times, respectively, and T_0 is the reference temperature. The electric field is computed by solving the Poisson equation

$$\nabla \cdot (\theta \mathbf{E}) = -\epsilon (n - N_D^+) \quad (7)$$

where θ is the dielectric permittivity and N_D^+ is the concentration of the ionized donor. In solving (1)-(3), electric field will be treated as a constant source term.

Remarks:

- i) A similar set of equations obeying the three conservation laws (1)-(3) can also be written for holes. In this paper, the analysis for well-posed boundary conditions will be presented for the electron particle system. The results presented will also hold for the hole particle system.
- ii) A similarity exists between the HD equations and equations of compressible Euler and Navier-Stokes [8]. It is interesting to note that the HD device equations are not identical to either the Euler equations because of the presence of heat conduction term or the Navier-Stokes equations because of the absence of viscous terms. Furthermore, the HD device equations contain very strong nonlinear source terms not commonly seen in fluid dynamics problems.

3 Primitive Variable Form

The hydrodynamic equations introduced in the previous section can be written in terms of primitive variables (n, u, T). The primitive variables are used to analyze the number of boundary conditions that need to be specified at the inflow and the outflow boundaries for a well-posed Initial Boundary Value Problem (IBVP). Let's first introduce the following variables and their definitions:

- i) $w = nm \left(c_v T + \frac{1}{2} |u|^2 \right)$ denotes the electron energy density, where m is the electron mass.
- ii) c_p, c_v are the specific heats at constant pressure and volume, respectively
- iii) $\gamma = \frac{c_p}{c_v}$ denotes the ratio of specific heats
- iv) $P = \frac{nk_b T}{m}$ denotes the electron pressure per unit mass.

It can be shown that the electron gas satisfies the perfect gas law with $\gamma = \frac{5}{3}$; the gas constant $R = \frac{k_b}{m}$; and the heat flux $q = -\kappa \nabla T$, where κ is defined by the Wiedemann-Franz law (see [8] and references therein).

The conservation laws can be rewritten in a primitive variable form using indicial notation as

$$\frac{\partial n}{\partial t} + \frac{\partial}{\partial x_i} (nu_i) = 0 \quad (8)$$

$$n \frac{\partial u_i}{\partial t} = F_i - nR \frac{\partial T}{\partial x_i} - RT \frac{\partial n}{\partial x_i} - nu_j \frac{\partial u_i}{\partial x_j} \quad (9)$$

$$n \frac{\partial T}{\partial t} = -nu_i \frac{\partial T}{\partial x_i} - (\gamma - 1) nT \frac{\partial u_i}{\partial x_i} + \frac{\partial}{\partial x_i} \left(\frac{\kappa (\gamma - 1)}{mR} \frac{\partial T}{\partial x_i} \right) - \frac{(\gamma - 1) u_i F_i}{R} + \frac{(\gamma - 1) F_{i+1}}{R} \quad (10)$$

where x_i ($i = 1, 2, 3$) denotes the spatial coordinates (x, y and z for $i = 1$ to 3, respectively),

$F_i = n \left[-\frac{\epsilon}{m} E_i - \frac{u_i}{\tau_p} \right]$, and $F_{i+1} = \frac{(w - w_0)}{m\tau_w}$. In the above equations repeated indices imply summation.

Equations (8)-(10) can be rewritten using matrix operators as follows:

$$\frac{\partial}{\partial t} \hat{U} = \hat{A}_i \frac{\partial}{\partial x_i} \hat{U} + \hat{K}_{ij} \frac{\partial^2}{\partial x_i \partial x_j} \hat{U} + \hat{F} \quad (11)$$

where \hat{U} denotes the primitive variables, \hat{A}_i denotes the advection matrices, \hat{K}_{ij} denotes the diffusion matrices and \hat{F} denotes the source vector consisting of the collision and electric field terms. The explicit definitions of the advection matrices are given below with $\hat{U} = \{T, n, u\}^T$

$$\hat{A}_1 = \begin{bmatrix} -u_1 & 0 & -(\gamma-1)T & 0 & 0 \\ 0 & -u_1 & -n & 0 & 0 \\ -R & -\frac{RT}{n} & -u_1 & 0 & 0 \\ 0 & 0 & 0 & -u_1 & 0 \\ 0 & 0 & 0 & 0 & -u_1 \end{bmatrix} \quad (12)$$

$$\hat{A}_2 = \begin{bmatrix} -u_2 & 0 & 0 & -(\gamma-1)T & 0 \\ 0 & -u_2 & 0 & -n & 0 \\ 0 & 0 & -u_2 & 0 & 0 \\ -R & -\frac{RT}{n} & 0 & -u_2 & 0 \\ 0 & 0 & 0 & 0 & -u_2 \end{bmatrix} \quad (13)$$

$$\hat{A}_3 = \begin{bmatrix} -u_3 & 0 & 0 & 0 & -(\gamma-1)T \\ 0 & -u_3 & 0 & 0 & -n \\ 0 & 0 & -u_3 & 0 & 0 \\ 0 & 0 & 0 & -u_3 & 0 \\ -R & -\frac{RT}{n} & 0 & 0 & -u_3 \end{bmatrix} \quad (14)$$

Note that \hat{A}_i are square but non-symmetric matrices. Similarly, the diffusion matrices can be expressed as $\hat{K}_{ij} = \hat{K}\delta_{ij}$ where δ_{ij} is the kronecker delta ($\delta_{ij} = 1$ for $i = j$ and $\delta_{ij} = 0$ for $i \neq j$) and

$$\hat{K} = \begin{bmatrix} \frac{\kappa(\gamma-1)}{nmR} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (15)$$

It is obvious that \hat{K}_{ij} are rank-deficient matrices.

Remark:

The system of equations given in (11) is referred to as a parabolic system when the diffusion matrices are positive definite and is generally termed as an incompletely parabolic system if the diffusion matrices are rank deficient. In the absence of the diffusion matrices, the system is hyperbolic. If the coefficient matrices are symmetric then the system is appropriately referred to as symmetric hyperbolic/parabolic/incompletely parabolic system.

4 Conditions for Well-Posedness

The literature on well-posedness for incompletely parabolic problems dates back to 1970's. Strikwerda's thesis [5] on well-posed boundary conditions for incompletely parabolic problems addressed several issues related to the necessary and sufficient conditions for the PDE systems of form (11) to be well-posed. This work also paved way for a number of studies addressing boundary conditions for several physical problems. Of notable interest is the one by Gustafson and Sundstrom [6], addressing the issue of well-posed boundary conditions for equations of fluid dynamics and shallow water. By following the work in these two references, we extend the concepts to study the proper boundary conditions for the HD device equations. We would like to emphasize that the HD equations can be considered intermediary between Euler and Navier-Stokes (NS) equations. To derive the number of boundary conditions that need to be imposed at inflow and outflow boundaries, we make use of several results reported in references [5] and [6]. Here, we briefly state the main theorems and definitions; interested readers are referred to the references for the proof of these theorems.

Definition 1: Let \hat{U}_0 be the initial conditions to (11). The system (11) is said to be well-posed if there is a constant C such that

$$\|\hat{U}\| \leq C (\|\hat{U}_0\| + \|\hat{F}\|) \quad (16)$$

Theorem 1 (Strikwerda [5] and Gustafson et. al [6]): Consider the incompletely parabolic system of partial differential equations given in (11) with constant coefficient matrices. The diffusion matrices \hat{K}_{ij} are rank deficient with some rank $r < n$, where n is the order of the square matrices \hat{A}_i and \hat{K}_{ij} . We

further assume that \hat{K}_{ij} can be represented (via some transformation) as

$$\hat{K}_{ij} = \begin{bmatrix} \hat{K}_{ij}^{(11)} & 0 \\ 0 & 0 \end{bmatrix} \quad (17)$$

so that \hat{A}_i are also rearranged accordingly as

$$\hat{A}_i = \begin{bmatrix} \hat{A}_i^{(11)} & \hat{A}_i^{(12)} \\ \hat{A}_i^{(21)} & \hat{A}_i^{(22)} \end{bmatrix} \quad (18)$$

where \hat{U} is partitioned as $\hat{U} = [\hat{U}_I, \hat{U}_{II}]^T$. For system (11) to be well posed, we require that the system

$$\frac{\partial \hat{U}_I}{\partial t} = \hat{K}_{ij}^{(11)} \frac{\partial^2 \hat{U}_I}{\partial x_i \partial x_j} \quad (19)$$

be parabolic and that the system

$$\frac{\partial \hat{U}_{II}}{\partial t} = \hat{A}_i^{(22)} \frac{\partial \hat{U}_{II}}{\partial x_i} \quad (20)$$

be strictly hyperbolic.

Theorem 2 (Strikwerda [5]): Consider the initial boundary value problem for the system (11) on a half space; i.e. $x_1 \geq 0$ and $-\infty < x_2, x_3 < \infty$ with constant coefficients and no lower order terms. For the system (11) to be well-posed the number of independent boundary conditions is given by $r + p$, where r is the rank of \hat{K}_{11} and p is the number of negative eigenvalues of $\hat{A}_1^{(22)}$.

Theorem 3 (Strikwerda [5]): Suppose the system (11) is approximated by a set of frozen coefficient matrices. If the approximated system (11) is well-posed, then system (11) is well-posed.

Remarks:

- i) In Theorem 1, it was assumed that \hat{K}_{ij} and \hat{A}_i undergo a particular transformation. This transformation can be easily ensured for HD equations by defining $\hat{U} = \{T, n, u\}^T$.

- ii) Gustafson and Sundstrom [6] have shown that the definition given for well-posedness in Theorem 1 is not very restrictive. They illustrated the problem using examples where the conditions stated in Theorem 1 are satisfied, but the solution has an exponential growth rate. However, such exponential growth rates are not possible for symmetrizable incompletely parabolic systems. Since the NS and HD equations can be symmetrized (see section 5), Theorem 1 applies to these equations.
- iii) Using the result in Theorem 2, our analysis will be performed for an inflow boundary parallel to the y-axis (or an electrical contact parallel to y-axis). The analysis and results apply analogously to inflow boundaries parallel to x- or z-axis.
- iv) With Theorem 3, the examination of well-posed boundary conditions can be restricted to constant coefficient systems, instead of the more general quasi-linear system of equations.

4.1 Number of independent conditions for contacts

The theorems cited above can be applied directly to determine the number of independent boundary conditions for the HD equations. In the following the analysis is performed on the equations for the general three-dimensional problem, and the results are analogously applicable for one- and two-dimensional problems. From the matrix definitions given in equations (12)-(15), it is clear that the rank of the diffusion matrix \hat{K}_{11} is one and the submatrix $\hat{A}_1^{(22)}$ of the advection matrix \hat{A}_1 is given as

$$\hat{A}_1^{(22)} = \begin{bmatrix} -u_1 & -n & 0 & 0 \\ \frac{RT}{n} & -u_1 & 0 & 0 \\ 0 & 0 & -u_1 & 0 \\ 0 & 0 & 0 & -u_1 \end{bmatrix} \quad (21)$$

According to Theorem 2, the number of boundary conditions can be determined by finding the number of negative eigenvalues of the above matrix. The four eigenvalues of $\hat{A}_1^{(22)}$ are

$$\begin{aligned} \lambda_{1,2} &= -u_1 \\ \lambda_3 &= -u_1 + c \\ \lambda_4 &= -u_1 - c \end{aligned} \quad (22)$$

where $c = \sqrt{RT}$ is the speed of sound. The number of boundary conditions can now be derived by classifying the inflow and outflow as either subsonic ($|u_1| < c$) or supersonic ($|u_1| > c$) flow:

1. Subsonic inflow ($c > u_1 > 0$): In this case three of the eigenvalues ($\lambda_1, \lambda_2, \lambda_4$) of $\hat{A}_1^{(22)}$ are negative. We thus need to specify a total of 4 boundary conditions. Comparing to the Euler and NS equations, we need 4 and 5 boundary conditions, respectively, for the inflow to ensure well-posedness of the system.

2. Subsonic outflow ($0 > u_1 > -c$): In this case there is only one negative eigenvalue (λ_4) in $\hat{A}_1^{(22)}$. Therefore, we need to specify a total of 2 boundary conditions. Comparing to the Euler and NS equations, we need 1 and 4 boundary conditions, respectively, for the outflow to ensure well-posedness of the system.

3. Supersonic inflow ($u_1 > c > 0$): In this case all four eigenvalues of $\hat{A}_1^{(22)}$ are negative. We thus need to specify 5 boundary conditions. The Euler and NS equations also require 5 boundary conditions for a well-posed system.

4. Supersonic outflow ($0 > -c > u_1$): In this case all eigenvalues of $\hat{A}_1^{(22)}$ are positive and we need to specify just 1 boundary condition. As for the Euler and NS equations, we need 0 and 4, boundary conditions, respectively, for the outflow to ensure well-posedness of the system.

Remarks:

- i) Table 1 summarizes the number of independent boundary conditions for one-, two- and three dimensional flows for the Euler, Navier-Stokes and HD equations.
- ii) The number of boundary conditions that need to be specified for the HD equations and that for the Euler or Navier-Stokes flow are not the same.
- iii) In general we can express the number of boundary conditions in terms of the number of primitive variables (i.e. the degree of freedom *ndof* per each node) as tabulated in Table 2. Note that $ndof = nsd + 2$, where *nsd* is the number of space dimensions equal to 1, 2, 3 for 1D, 2D and 3D problems respectively.

Table 1: Number of independent boundary conditions

Type of flow		Euler	NS	HD
one dimensional flow	<i>subsonic inflow</i>	2	3	2
	<i>subsonic outflow</i>	1	2	2
	<i>supersonic inflow</i>	3	3	3
	<i>supersonic outflow</i>	0	2	1
two dimensional flow	<i>subsonic inflow</i>	3	4	3
	<i>subsonic outflow</i>	1	3	2
	<i>supersonic inflow</i>	4	4	4
	<i>supersonic outflow</i>	0	3	1
three dimensional flow	<i>subsonic inflow</i>	4	5	4
	<i>subsonic outflow</i>	1	4	2
	<i>supersonic inflow</i>	5	5	5
	<i>supersonic outflow</i>	0	4	1

Table 2: Summary of independent boundary conditions for 1-, 2-, and 3D flows

Type of flow	Euler	NS	HD
<i>subsonic inflow</i>	<i>ndof-1</i>	<i>ndof</i>	<i>ndof-1</i>
<i>subsonic outflow</i>	1	<i>ndof-1</i>	2
<i>supersonic inflow</i>	<i>ndof</i>	<i>ndof</i>	<i>ndof</i>
<i>supersonic outflow</i>	0	<i>ndof-1</i>	1

5 Symmetric Forms and Energy Estimates

In this section we apply the classical energy method to show well-posedness for symmetrizable incompletely parabolic systems. We discuss two different approaches to symmetrize the HD equations and subsequently derive energy estimates. In the first approach the HD equations are symmetrized by retaining the primitive variables as the basic variables. In the second approach, generalized entropy functions are employed to symmetrize the system of equations. In the latter approach, the basic variables are different from the primitive variables and will be referred to as entropy variables. There are fundamental advantages to the entropy variable formulation, which has been employed in the development of a finite element formulation for the HD equations [8].

5.1 Symmetric form employing primitive variables

The HD system given in (11) can be symmetrized by multiplying the equation with a symmetric positive definite matrix \hat{R} given as follows:

$$\hat{R} = \begin{bmatrix} \frac{nR}{T(\gamma-1)} & 0 & 0 & 0 & 0 \\ 0 & \frac{RT}{n} & 0 & 0 & 0 \\ 0 & 0 & n & 0 & 0 \\ 0 & 0 & 0 & n & 0 \\ 0 & 0 & 0 & 0 & n \end{bmatrix} \quad (23)$$

Equation (11) can thus be written as

$$\hat{R} \frac{\partial}{\partial t} \hat{U} = \hat{R} \hat{A}_i \frac{\partial}{\partial x_i} \hat{U} + \hat{R} \hat{K}_{ij} \frac{\partial^2}{\partial x_i \partial x_j} \hat{U} + \hat{R} \hat{F} \quad (24)$$

It can be easily verified that the coefficient matrices $\hat{R} \hat{A}_i$ and $\hat{R} \hat{K}_{ij}$ are symmetric. As reported in [6], for compressible NS equations, equation (24) can be rewritten in the following general form

$$\hat{R} \frac{\partial}{\partial t} \hat{U} = \left(\hat{R} \hat{A}_i - \frac{\partial}{\partial x_j} (\hat{R} \hat{K}_{ij}) \right) \frac{\partial}{\partial x_i} \hat{U} + \frac{\partial}{\partial x_j} \left(\hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_i} \hat{U} \right) + \hat{R} \hat{F} \quad (25)$$

Equation (25) is called symmetric if the coefficient matrices $\hat{R} \hat{K}_{ij}$ and $\hat{R} \hat{A}_i - \frac{\partial}{\partial x_j} (\hat{R} \hat{K}_{ij})$ are symmetric.

The well-posedness for equation (25) can be demonstrated using the classical energy method. Assuming that \hat{A}_i and \hat{K}_{ij} are constant coefficient matrices and that the deviations between \hat{U} and the exact solution \hat{U} are small, we obtain the following variational equation

$$\hat{R} \frac{\partial}{\partial t} \hat{U}' = \left(\hat{R} \hat{A}_i - \frac{\partial}{\partial x_j} (\hat{R} \hat{K}_{ij}) \right) \frac{\partial}{\partial x_i} \hat{U}' + \frac{\partial}{\partial x_j} \left(\hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_i} \hat{U}' \right) + \hat{R} \hat{F}' \quad (26)$$

Noting that

$$\frac{\partial}{\partial t} \left(\hat{U}'^T \hat{R} \hat{U}' \right) = \frac{\partial \hat{U}'^T}{\partial t} \hat{R} \hat{U}' + \hat{U}'^T \frac{\partial \hat{R}}{\partial t} \hat{U}' + \hat{U}'^T \hat{R} \frac{\partial}{\partial t} \hat{U}' \quad (27)$$

and substituting (26) in (27), we obtain the following energy growth equation

$$\begin{aligned} \frac{\partial}{\partial t} \left(\hat{U}'^T \hat{R} \hat{U}' \right) &= \frac{\partial}{\partial x_i} \left(\hat{U}'^T \hat{R} \hat{A}_i \hat{U}' \right) + \frac{\partial}{\partial x_i} \left(\hat{U}'^T \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_j} \hat{U}' + \frac{\partial \hat{U}'^T}{\partial x_j} \hat{R} \hat{K}_{ij} \hat{U}' \right) - \frac{\partial \hat{U}'^T}{\partial x_i} \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_j} \hat{U}' - \\ &\quad \frac{\partial \hat{U}'^T}{\partial x_j} \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_i} \hat{U}' + \hat{U}'^T \left(\frac{\partial \hat{R}}{\partial t} - \frac{\partial}{\partial x_i} \left(\hat{R} \hat{A}_i - \frac{\partial}{\partial x_j} \hat{R} \hat{K}_{ij} \right) \right) \hat{U}' + \hat{F}'^T \hat{R} \hat{U}' + \hat{U}'^T \hat{R} \hat{F}' \end{aligned} \quad (28)$$

Integrating over the domain, Ω , and applying divergence theorem, one obtains

$$\begin{aligned} \frac{\partial}{\partial t} \int_{\Omega} (\hat{U}^T \hat{R} \hat{U}) d\Omega &= \int_{\Gamma} (\hat{U}^T \hat{R} \hat{A}_i \hat{U}) n_i d\Gamma + 2 \int_{\Gamma} (\hat{U}^T \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_j} \hat{U}) n_i d\Gamma - \int_{\Omega} \frac{\partial \hat{U}^T}{\partial x_i} \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_j} \hat{U} d\Omega - \\ &\int_{\Omega} \frac{\partial \hat{U}^T}{\partial x_j} \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_i} \hat{U} d\Omega + \int_{\Omega} \hat{U}^T \left(\frac{\partial}{\partial t} \hat{R} - \frac{\partial}{\partial x_i} \left(\hat{R} \hat{A}_i - \frac{\partial}{\partial x_j} \hat{R} \hat{K}_{ij} \right) \right) \hat{U} d\Omega + \int_{\Omega} \hat{F}^T \hat{R} \hat{U} + \hat{U}^T \hat{R} \hat{F} d\Omega \end{aligned} \quad (29)$$

where Γ denotes the boundary of the physical domain and n_i denotes the unit outward normal. From the definition of \hat{K}_{ij} , we can establish that

$$\begin{aligned} \int_{\Omega} \frac{\partial \hat{U}^T}{\partial x_i} \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_j} \hat{U} d\Omega &\geq 0 \\ \int_{\Omega} \frac{\partial \hat{U}^T}{\partial x_j} \hat{R} \hat{K}_{ij} \frac{\partial}{\partial x_i} \hat{U} d\Omega &\geq 0 \end{aligned} \quad (30)$$

Choosing $\mathbf{n} = (-1, 0, 0)$ (which by no means is a simplifying assumption¹), where x -axis points in the direction of inward normal and y and z planes are tangential to Γ , we obtain

$$\frac{\partial}{\partial t} \int_{\Omega} (\hat{U}^T \hat{R} \hat{U}) d\Omega \leq \int_{\Omega} \hat{U}^T \left(\frac{\partial}{\partial t} \hat{R} - \frac{\partial}{\partial x_i} \left(\hat{R} \hat{A}_i - \frac{\partial}{\partial x_j} \hat{R} \hat{K}_{ij} \right) \right) \hat{U} d\Omega + \int_{\Omega} \hat{F}^T \hat{R} \hat{U} + \hat{U}^T \hat{R} \hat{F} d\Omega \quad (31)$$

with the assumption that

$$\int_{\Gamma} \left(\hat{U}^T \hat{R} \hat{A}_1 \hat{U} + 2 \hat{U}^T \hat{R} \hat{K}_{11} \frac{\partial}{\partial x_1} \hat{U} \right) d\Gamma \geq 0 \quad (32)$$

Defining $\|\hat{U}\|^2 = \int_{\Omega} \hat{U}^T \hat{R} \hat{U} d\Omega$ which is equivalent to the L^2 norm $\int_{\Omega} |\hat{U}|^2 d\Omega$ [7], and noting that \hat{R}

and $\hat{R} \hat{A}_i - \frac{\partial}{\partial x_j} \hat{R} \hat{K}_{ij}$ are bounded matrices, equation (31) gives the following growth equation

$$\frac{\partial \|\hat{U}\|^2}{\partial t} \leq 2C \|\hat{U}\|^2 + 2\|\hat{U}\| \|\hat{F}\| \quad (33)$$

where $C > 0$ is some constant. Equation (33) gives the following estimate for well-posedness

$$\|\hat{U}(t)\| \leq e^{Ct} \|\hat{U}(0)\| + \|\hat{F}\|_{\Omega \times [0, t]} \quad (34)$$

where $\Omega \times [0, t]$ denotes an integral over space and time.

¹ Note that we can select any arbitrary normal vector since the HD equations are rotationally invariant and we can consider a moving coordinate frame

The boundary conditions can now be chosen such that equation (32) is satisfied. The expression for energy estimate is given by

$$\|\hat{U}\|^2 = \int_{\Omega} n \sum_{i=1}^{nsd} u_i'^2 + nRT \left(\left(\frac{n'}{n} \right)^2 + (\gamma-1)^{-1} \left(\frac{T'}{T} \right)^2 \right) d\Omega \quad (35)$$

From the above equation it is clear that $\|\hat{U}\|^2$ is positive and can be used as a well-defined quantity for energy estimate. The boundary conditions should thus be chosen to satisfy the following inequality

$$-nu_1 \left(\sum_{i=1}^{nsd} u_i'^2 + \frac{RT}{(\gamma-1)} \left(\frac{T'}{T} \right)^2 + RT \left(\frac{n'}{n} \right)^2 \right) - 2RTn'u_1' - 2nRTu_1' + 2\frac{\kappa}{m}TT^{-1}\frac{\partial}{\partial x}T \geq 0 \quad (36)$$

Examples for well-posed boundary conditions based on this inequality are derived in section 6.

5.2 Symmetric form employing entropy variables

In our formulation and implementation of HD equations, entropy variables are employed instead of the primitive or conservation variables [8]. The HD equations given in (1)-(3) can be written in a system form using conservation variables as follows:

$$\frac{\partial U}{\partial t} + \frac{\partial F_i^E}{\partial x_i} = \frac{\partial F_i^H}{\partial x_i} + F \quad (37)$$

where $U = \{n, nu, ne_{tot}\}^T$ denotes the vector of conservation variables,

$F_i^E = \{nu_i, nu_iu_1 + P\delta_{1i}, nu_iu_2 + P\delta_{2i}, nu_iu_3 + P\delta_{3i}, nu_ie_{tot} + Pu_i\}^T$ denotes the Euler flux vector,

$F_i^H = \{0, 0, 0, 0, q_i\}^T$ denotes the vector of diffusive flux (or heat conduction) and F is the source

vector containing the remaining terms from equations (1)-(3). Equation (37) can be rewritten in a quasi-linear form as

$$\frac{\partial U}{\partial t} + A_i \frac{\partial U}{\partial x_i} = \frac{\partial}{\partial x_i} \left(K_{ij} \frac{\partial U}{\partial x_j} \right) + F \quad (38)$$

where $A_i = \frac{\partial F_i^E}{\partial U}$ and $K_{ij} \frac{\partial U}{\partial x_j} = F_i^H$. The matrices A_i do not possess the properties of symmetry or positiveness. A symmetric form of equation (38) can be obtained by a change of variables using gen-

eralized entropy functions [9]. By considering generalized entropy functions of the form $\mathcal{H} = -ns$, where s is the thermodynamic entropy per unit mass, and introducing a change of variables defined by

$$V^T = \frac{\partial \mathcal{H}}{\partial U} \quad (39)$$

a symmetrized system of the form

$$\tilde{A}_0 \frac{\partial V}{\partial t} = \tilde{A}_i \frac{\partial V}{\partial x_i} + \frac{\partial}{\partial x_i} \left(\tilde{K}_{ij} \frac{\partial V}{\partial x_j} \right) + F \quad (40)$$

is obtained. In the above equation, $\tilde{A}_0 = \frac{\partial U}{\partial V}$ is symmetric and positive-definite, $\tilde{A}_i = -A_i \tilde{A}_0$ is symmetric and $\tilde{K}_{ij} = K_{ij} \tilde{A}_0$.

In our work we select a thermodynamic entropy of the form $s = c_v \ln \left[\frac{P}{n^\gamma} \right]$ to obtain the symmetric form (40). Equation (40) can be compared with similar forms derived for primitive variable approach in equations (24) and (25). Note, however, that the terms in the coefficient matrices are different. The following definitions for matrix coefficients will be used in deriving conditions for well-posedness.

$$\tilde{A}_0 = \frac{n}{R} \begin{bmatrix} 1 & u_1 & u_2 & u_3 & h+k-RT \\ u_1^2 + RT & u_1 u_2 & u_1 u_3 & u_1 (h+k) \\ & u_2^2 + RT & u_2 u_3 & u_2 (h+k) \\ & & u_3^2 + RT & u_3 (h+k) \\ & & & (h+k)^2 - hRT \end{bmatrix} \quad (41)$$

$$\tilde{A}_1 = -\frac{n}{R} \begin{bmatrix} u_1 & u_1^2 + RT & u_1 u_2 & u_1 u_3 & u_1 (h+k) \\ u_1 (u_1^2 + RT) & u_2 (u_1^2 + RT) & u_3 (u_1^2 + RT) & (h+k) (u_1^2 + RT) + u_1^2 RT \\ & u_1 (u_2^2 + RT) & u_1 u_2 u_3 & u_1 u_2 (h+k+RT) \\ & & u_1 (u_3^2 + RT) & u_1 u_3 (h+k+RT) \\ & & & u_1 ((h+k)^2 + (h+2k)RT) \end{bmatrix} \quad (42)$$

where $h = c_p T$ and $k = \frac{|u|^2}{2}$. The diffusion matrices are given as $\tilde{K}_{ij} = \tilde{K} \delta_{ij}$, where

$$\tilde{K} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\kappa T^2}{m} \end{bmatrix} \quad (43)$$

For the definitions of all other advection and diffusion matrices, the readers are referred to [8]. The entropy variable vector V is given as

$$V = \frac{1}{T} \left[\mu - \frac{|u|^2}{2}, u_1, u_2, u_3, -1 \right]^T \quad (44)$$

where $\mu = \frac{RT}{(\gamma-1)} + \frac{P}{n} - Ts$ is the specific chemical potential.

5.2.1 Finite element formulation

The details on a finite element formulation for the hydrodynamic device equations are given in [8]. Here, we briefly review the finite element formulation since the development of the natural boundary conditions requires information about the boundary integral present in the weak form.

In our approach, a space-time Galerkin/least-squares finite element method is employed to solve the HD equations. The time-dependent hydrodynamic equations are solved for the steady-state solution by employing a time marching algorithm. Within each time interval a Galerkin/least-squares finite element method is employed in space and a time-discontinuous Galerkin method is employed in time. To provide stability to the Galerkin finite element method, the time-discontinuous Galerkin method is augmented by adding terms of a least-squares type. The time-discontinuous Galerkin finite element method can be obtained by multiplying the strong form (Equation (37) or (40)) by a test function W and integrating over the space-time slab. This step leads to the following equation:

$$\begin{aligned} \int_{t_n}^{t_{n+1}} \int_{\Omega} \left[-\frac{\partial W}{\partial t} U(V) - \frac{\partial W}{\partial x_i} F_i^E + \frac{\partial W}{\partial x_i} F_i^H - WF \right] d\Omega dt + \int_{\Omega} (W(t_{n+1}) U(t_{n+1}) - W(t_n) U(t_n)) d\Omega \\ + \int_{t_n}^{t_{n+1}} \int_{\Gamma} [W(F_i^E - F_i^H)] n_i d\Gamma dt = 0 \end{aligned} \quad (45)$$

where $[t_n, t_{n+1}]$ is the time slab, Ω is the domain, Γ is the boundary and n_i is the unit outward normal. The boundary integral shown in equation (45) is the main equation for the derivation of natural boundary conditions. It should however be noted that the time-discontinuous Galerkin method is not a stable method for solving HD equations [13], in contradictory to the notion that Galerkin methods are convergent for symmetrized HD equations [12].

5.2.2 Energy estimate for entropy variable form

A variational form for equation (40), by locally freezing the matrix operators, takes the form

$$\tilde{A}_0 \frac{\partial}{\partial t} V = \tilde{A}_i \frac{\partial}{\partial x_i} V + \frac{\partial}{\partial x_i} \left(\tilde{K}_{ij} \frac{\partial}{\partial x_j} V \right) + F \quad (46)$$

Noting that

$$\frac{\partial}{\partial t} \left(V^T \tilde{A}_0 V \right) = \frac{\partial V^T}{\partial t} \tilde{A}_0 V + V^T \frac{\partial \tilde{A}_0}{\partial t} V + V^T \tilde{A}_0 \frac{\partial}{\partial t} V \quad (47)$$

and substituting (46) into (27), we obtain the following energy growth equation

$$\begin{aligned} \frac{\partial}{\partial t} \left(V^T \tilde{A}_0 V \right) &= \frac{\partial}{\partial x_i} \left(V^T \tilde{A}_i V \right) + \frac{\partial}{\partial x_i} \left(V^T \tilde{K}_{ij} \frac{\partial}{\partial x_j} V + \frac{\partial V^T}{\partial x_j} \tilde{K}_{ij} V \right) - \frac{\partial V^T}{\partial x_i} \tilde{K}_{ij} \frac{\partial}{\partial x_j} V - \\ &\quad \frac{\partial V^T}{\partial x_j} \tilde{K}_{ij} \frac{\partial}{\partial x_i} V + V^T \left(\frac{\partial \tilde{A}_0}{\partial t} - \frac{\partial \tilde{A}_i}{\partial x_i} \right) V + F^T V + V^T F \end{aligned} \quad (48)$$

Integrating over the domain, Ω gives

$$\begin{aligned} \frac{\partial}{\partial t} \int_{\Omega} V^T \tilde{A}_0 V d\Omega &= \int_{\Gamma} V^T \tilde{A}_i V n_i d\Gamma + 2 \int_{\Gamma} V^T \tilde{K}_{ij} \frac{\partial}{\partial x_j} V n_i d\Gamma - \int_{\Omega} \frac{\partial V^T}{\partial x_i} \tilde{K}_{ij} \frac{\partial}{\partial x_j} V d\Omega - \\ &\quad \int_{\Omega} \frac{\partial V^T}{\partial x_j} \tilde{K}_{ij} \frac{\partial}{\partial x_i} V d\Omega + \int_{\Omega} V^T \left(\frac{\partial \tilde{A}_0}{\partial t} - \frac{\partial \tilde{A}_i}{\partial x_i} \right) V d\Omega + \int_{\Omega} F^T V + V^T F d\Omega \end{aligned} \quad (49)$$

Using the definition of \tilde{K}_{ij} , it can be shown that

$$\begin{aligned} \int_{\Omega} \frac{\partial V^T}{\partial x_i} \tilde{K}_{ij} \frac{\partial}{\partial x_j} V d\Omega &\geq 0 \\ \int_{\Omega} \frac{\partial V^T}{\partial x_j} \tilde{K}_{ij} \frac{\partial}{\partial x_i} V d\Omega &\geq 0 \end{aligned} \quad (50)$$

Choosing $n = (-1, 0, 0)$ and assuming

$$\int_{\Gamma} V^T \tilde{A}_1 V d\Gamma + 2 \int_{\Gamma} V^T \tilde{K}_{11} \frac{\partial}{\partial x_1} V d\Gamma \geq 0 \quad (51)$$

we obtain the following inequality

$$\frac{\partial}{\partial t} \int_{\Omega} V^T \tilde{A}_0 V d\Omega \leq \int_{\Omega} V^T \left(\frac{\partial \tilde{A}_0}{\partial t} - \frac{\partial \tilde{A}_i}{\partial x_i} \right) V d\Omega + \int_{\Omega} F^T V + V^T F d\Omega \quad (52)$$

Defining, $\|V\|^2 = \int_{\Omega} V^T \tilde{A}_0 V d\Omega$ and noting that \tilde{A}_0 and \tilde{A}_i are bounded matrices, we obtain the following growth equation

$$\frac{\partial \|V\|^2}{\partial t} \leq 2C_1 \|V\|^2 + 2C_2 \|V\| \|F\| \quad (53)$$

where $C_1, C_2 > 0$ are constants. An estimate for well-posedness can be derived from equation (53) and takes the following form

$$\|V(t)\| \leq e^{C_1 t} \|V(0)\| + C_2 \|F\|_{\Omega \times [0, t]} \quad (54)$$

The boundary conditions for well-posedness should be chosen to satisfy equation (51). Employing the following definition for V

$$V = \left[-\frac{u_i}{T} u'_i + \frac{|u|^2}{2T^2} T + R \frac{n'}{n} - c_v \frac{T}{T}, \frac{Tu'_1 - u_1 T}{T^2}, \frac{Tu'_1 - u_1 T}{T^2}, \frac{Tu'_3 - u_3 T}{T^2}, \frac{T}{T^2} \right]^T \quad (55)$$

and \tilde{A}_0 given in equation (41), we obtain the following expression for the energy estimate

$$\|V\|^2 = \int_{\Omega} \frac{1}{T} \left[n \sum_{i=1}^{nsd} u_i'^2 + nRT \left(\left(\frac{n'}{n} \right)^2 + (\gamma - 1)^{-1} \left(\frac{T}{T} \right)^2 \right) \right] d\Omega \quad (56)$$

It is interesting to note that the energy measure for the entropy variable approach differs from the energy measure for the primitive variable approach (see equation (35)) by the coefficient $\frac{1}{T}$.

Using \tilde{A}_1 and \tilde{K}_{11} given in equations (42) and (43) respectively, equation (51) gives the following condition for selecting boundary conditions

$$\frac{1}{T} \left[-nu_1 \left(\sum_{i=1}^{nsd} u_i'^2 + \frac{RT}{(\gamma-1)} \left(\frac{T}{T} \right)^2 + RT \left(\frac{n'}{n} \right)^2 \right) - 2RTn'u'_1 - 2nRTu'_1 + 2\frac{\kappa}{m}TT^{-1} \frac{\partial}{\partial x}(T) \right] \geq 0 \quad (57)$$

For T being a positive quantity, this relation is identical to the one obtained in equation (36).

Remarks:

- i) In the limit of negligible heat conduction, equation (57) reduces to the well-posedness condition for Euler equations.
- ii) The expression given in [6] for Navier-Stokes equations reduces to equation (57) in the absence of the viscous terms, verifying that equation (57) is indeed the condition for well-posedness of hydrodynamic equations.

6 Boundary conditions for contacts

The boundary conditions for the HD equations are imposed by satisfying the positivity conditions derived in equations (36) and (57). These two inequalities are essentially the same in that satisfying one inequality would also satisfy the other. In this section boundary conditions are derived that satisfy the inequality given in equation (36). For each of the four cases discussed before i.e. subsonic/supersonic inflow and subsonic/supersonic outflow, we derive a set(s) of boundary conditions and show that these boundary conditions satisfy the inequality (36).

1. Subsonic inflow ($c > u_1 > 0$)

From table 2 we need to specify 2, 3 and 4 boundary conditions respectively for 1D, 2D and 3D, respectively. One set of possible boundary conditions are summarized below

1D: $nu_1 = g_1$ and $T = g_2$

2D: $nu_1 = g_1, u_2 = g_2$ and $T = g_3$

3D: $nu_1 = g_1, u_2 = g_2, u_3 = g_3$ and $T = g_4$

where g_i denotes some prescribed value for the quantity to be specified. In the following we verify that the boundary conditions indeed satisfy the inequalities of (36) (or (57)). The prescribed boundary conditions would mean $u'_2 = u'_3 = T' = 0$. Substituting these in equation (36) (or equation (57)) would make the left hand side (*lhs*) of the inequality as

$$lhs = -nu_1 \left(u_1'^2 + RT \left(\frac{n'}{n} \right)^2 \right) - 2RTu'_1 n' \quad (58)$$

The boundary condition $nu_1 = g_1$ gives $\frac{n'}{n} = -\frac{u'_1}{u_1}$. Thus, we get

$$lhs = \frac{u_1^2}{g_1} \left(-u_1^2 + c^2 \right) \quad (59)$$

since the flow is subsonic, $lhs > 0$, thereby satisfying the inequality. The boundary conditions for 1D and 2D cases can be verified in a similar manner.

Another set of boundary conditions that can also be specified for subsonic inflow stems from Schottky barriers. In this type of boundary condition the normal component of current is related to the concentration [4]. For electrons, this condition is given as

$$-nu_1 = v_{th} (n - n_0) \quad (60)$$

where v_{th} is the thermionic velocity and n_0 is the equilibrium concentration. Using this condition, the second set of boundary conditions can be summarized as follows:

$$1D: u_1 = -v_{th} \left(1 - \frac{n_0}{n} \right) \text{ and } T = g_2$$

$$2D: u_1 = -v_{th} \left(1 - \frac{n_0}{n} \right), u_2 = g_2 \text{ and } T = g_3$$

$$3D: u_1 = -v_{th} \left(1 - \frac{n_0}{n} \right), u_2 = g_2, u_3 = g_3 \text{ and } T = g_4$$

For these boundary conditions, it can be shown that the inequality in equation (36) (or equation (57)) would be satisfied for the following condition

$$0 \leq u_1 \leq 2 \left(\frac{c^2 g}{g^2 + c^2} \right) \quad (61)$$

where $g = \frac{v_{th} n_0}{n}$. The second set of boundary conditions are often preferred over the first set for device simulation as the quantity nu_1 is not known.

It is to be observed that prescribing n , T and the tangential components of velocity (for multi-dimensional flows) are not well-posed boundary conditions, even though these are the commonly employed boundary conditions. We do not suggest that the boundary conditions discussed above (and hereafter) are by any means complete. For instance, in the case of a high level injection of a diode, none of the above sets of boundary conditions seem to be suitable. Development of a set of proper boundary conditions for such a device remains a subject for further investigation.

2. Subsonic outflow ($0 > u_1 > -c$)

For subsonic outflow, regardless of the space dimension of the problem, we need to specify two bound-

ary conditions. The inequalities can be satisfied by choosing one of the following three sets of boundary conditions.

1: $n = g_1$ and $T = g_2$.

2: $u_1 = -v_{th} \left(1 - \frac{n_0}{n}\right)$ and $T = g_2$

3: $u_1 = g_3$ and $\frac{\partial T}{\partial x} = g_4$.

In semiconductor device simulation, inflow velocity u_1 is typically not known. So the first two sets of boundary conditions are preferred over the third one. For the first set of boundary conditions the inequality is satisfied, i.e

$$-nu_1 \left(\sum_{i=1}^{nsd} u_i'^2 + \frac{RT}{(\gamma-1)} \left(\frac{T}{T}\right)^2 + RT \left(\frac{n'}{n}\right)^2 \right) > 0 \quad (62)$$

since $u_1 < 0$ and the quantity inside the parenthesis is positive. In the second set of boundary conditions (again based on Schottky barrier), a velocity boundary condition similar to the one suggested for subsonic inflow is employed. In this case the inequality takes the form

$$-nu_1 \left(\sum_{i=1}^{nsd} u_i'^2 + \frac{RT}{(\gamma-1)} \left(\frac{T}{T}\right)^2 + RT \left(\frac{n'}{n}\right)^2 \right) + \frac{2RTv_s n_0}{n^2} n'^2 \geq 0 \quad (63)$$

since $u_1 < 0$. Note that for this set of boundary conditions no limit is placed on the inflow velocity u_1 .

Commonly employed boundary conditions for 2D simulations (assume the contact placement is parallel to x-axis) are $n = g_1$, $u_2 = 0$ and $T = g_3$. Based on the above development we may say that this set of boundary conditions is an overspecification.

3. Supersonic inflow ($u_1 > c > 0$)

For supersonic inflow we need to specify 3, 4 and 5 boundary conditions for 1D, 2D and 3D problems, respectively. The number of conditions requires that all the basic nodal variables need to be specified. Thus we have the following set of boundary conditions

$$T = g_{nsd+2}, n = g_1, \text{ and } u_i = g_{i+1} \text{ where } i = 1, nsd$$

The boundary conditions mentioned here pose an interesting physical question. As noted earlier, the inflow velocity u_1 is typically not known. However, since the flow is supersonic we may impose that the inflow velocity cannot be greater than the saturation velocity. Alternatively, any other set of boundary conditions that satisfies the inequality (36) are also applicable. It is possible to develop better boundary conditions and this is a subject for further investigation. For the boundary conditions speci-

fied above, the inequality (36) is identically equal to zero. It should be mentioned that in semiconductor device simulation, supersonic inflow boundaries are rarely encountered.

4. Supersonic outflow ($0 > -c > u_1$)

Independent of the space dimension, only one boundary condition needs to be specified for this case.

Valid boundary conditions include setting $\frac{\partial T}{\partial x} = g_1$ or $T = g_2$. In this case the inequality takes the form

$$lhs = -nu_1 \left(\sum_{i=1}^{nsd} u_i'^2 + \frac{RT}{(\gamma-1)} \left(\frac{T}{T} \right)^2 + RT \left(\frac{n'}{n} \right)^2 \right) - 2RTn'u'_1 - 2nRTu'_1 \quad (64)$$

This equation can be rewritten as:

$$lhs = -nu_1 \left(\sum_{i=2}^{nsd} u_i'^2 \right) - \frac{u_1 n RT}{\gamma(\gamma-1)} \left[\left(\frac{T}{T} - (\gamma-1) \frac{n'}{n} \right)^2 \right] + \frac{1}{2} (-u_1 - c) n \left[u'_1 + \sqrt{\frac{RT}{\gamma}} \left(\frac{T}{T} + \frac{n'}{n} \right) \right]^2 + \frac{1}{2} (-u_1 + c) n \left[u'_1 - \sqrt{\frac{RT}{\gamma}} \left(\frac{T}{T} + \frac{n'}{n} \right) \right]^2 \quad (65)$$

For the current case $u_1 < 0$ and both $(-u_1 - c)$ and $(-u_1 + c)$ are positive, the inequality of equation (36) is thus satisfied.

Remarks:

- i) The examples presented for inflow and outflow boundaries and subsonic/supersonic cases are only some of several possible sets of boundary conditions. The examples discussed have either physical or mathematical basis and can easily be implemented.
- ii) Mixed type of boundary conditions (involving the quantity and its derivative) are among the feasible sets of boundary conditions. Reference [6] has some examples of this type for Euler and Navier-Stokes equations. Examples involving mixed type of boundary conditions are not presented here since they are usually more difficult to implement.
- iii) In practice, simulations are performed without verifying the well-posedness of the boundary conditions. If stable numerical schemes are employed, exponential growth in the solution can be avoided. However, where possible it is highly recommended that well-posed boundary conditions be specified to avoid steep gradients in the solution and to ensure the convergence behavior of the numerical scheme.

7 Natural boundary conditions

Artificial boundaries and interfaces to insulating material are typically specified by natural boundary conditions. Admissible natural boundary conditions can be extracted from the integrals present in the weak form of the finite element formulations. Unlike the boundary conditions discussed for contacts (which are generally termed as essential or Dirichlet boundary conditions) the natural boundary conditions will be imposed weakly and hence are often referred to as weak boundary conditions. The time boundary integral in the variational equation is given by

$$\int_{t_n}^{t_{n+1}} \int_{\Gamma} W \left(-F_i^E(V) + F_i^H(V) \right) n_i \, d\Gamma \, dt \quad (66)$$

Substituting the definitions for fluxes, we obtain

$$\int_{t_n}^{t_{n+1}} \int_{\Gamma} W^h \left(-n u_n \begin{bmatrix} 1 \\ u_1 \\ u_2 \\ u_3 \\ e + \frac{k_b T}{m} \end{bmatrix} - P \begin{bmatrix} 0 \\ \delta_{1n} \\ \delta_{2n} \\ \delta_{3n} \\ 0 \end{bmatrix} - q_n \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right) d\Gamma \, dt \quad (67)$$

where

$$u_n = u_i n_i \quad (68)$$

$$\delta_{in} = \delta_{ij} n_j \quad (69)$$

$$q_n = q_i n_i \quad (70)$$

From Equation (66), we can extract the following natural boundary conditions:

- i) normal mass flux or current per unit charge, $n u_n = h^c$.
- ii) carrier pressure per unit mass, $P = n k_b T / m = h^p$.
- iii) normal heat flux, $q_n = h^q$.

Remarks:

- i) For boundaries that act as interfaces to insulating material, a natural boundary condition can be prescribed for vanishing current i.e. $n u_n = 0$. This condition can also be specified through a Dirichlet boundary condition by prescribing zero normal velocity to the boundary. For example,

if a boundary that acts as interface is aligned along the x-axis, zero current flux along the normal direction can be specified by imposing $u_2 = 0$ (vertical velocity). Heat flux cannot be considered

negligible for interfaces to insulating material owing to physical reasons. Therefore, $\frac{\partial T}{\partial n} = 0$ cannot be treated as a proper boundary condition as this leads to vanishing heat flux.

- ii) Any physically reasonable boundary conditions can be specified on the artificial boundaries.

They are typically treated by employing vanishing currents ($nu_n = 0$) and heat fluxes ($\frac{\partial T}{\partial n} = 0$).

8 Example

The boundary conditions discussed in this paper are studied on a two-dimensional MESFET device. The MESFET device shown in Figure 1 consists of a barrier junction at the input that acts as a control electrode (or gate), and two ohmic contacts, described as source and drain electrodes, through which the output current flows. The source contact acts as an inflow boundary and the drain contact acts as an outflow boundary. The device is a special form of a junction field-effect transistor (JFET).

The three terminal device is $0.6\mu\text{m}$ long along the x-direction and $0.2\mu\text{m}$ wide along the y-direction. The contacts are placed on the top portion of the geometry. The source and drain contacts are approximately $0.1\mu\text{m}$ long and the gate contact is approximately $0.2\mu\text{m}$ long. The source and the drain contacts are separated from the gate contact by approximately $0.1\mu\text{m}$. The substrate of the device is doped n-type with a doping value of $1.0 \times 10^{17}/\text{cm}^3$. The two n^+ regions shown in Figure 2 are approximately of size $0.1\mu\text{m} \times 0.05\mu\text{m}$. The doping value in these regions is $3.0 \times 10^{17}/\text{cm}^3$ with abrupt junctions between n^+ and n boundaries.

A uniform mesh consisting of 3072 nodes and 2945 elements is used with 95 elements placed along the x-direction and 31 elements placed along the y-direction. The boundary conditions used for this experiment are summarized as follows:

- i) for source (h-g), $n = 3.0 \times 10^{17}/\text{cm}^3$, $u = 0$ cm/s, $T = 300$ K, and drain (d-c), $n = 3.0 \times 10^{17}/\text{cm}^3$, $T = 300$ K, and $\psi = \psi_b + \psi_{\text{appl}}$
- ii) for gate contact (f-e), $n = n_g$, $u = 0$ cm/s and $T = 300$ K, and $\psi = v_{g\text{eff}} = \psi_b - \psi_{\text{gappl}}$
- iii) on all other boundaries, $J_n = nu_n = 0$

The variable n_g denotes the concentration prescribed on the gate contact. The results for this experiment are shown in Figures 2 and 3. Figure 2 shows the electron concentration and Figure 3 shows the electron temperature. To keep our discussion concise, other variables such as the electron velocities, potential and electric fields are not plotted here but can be found in [11].

In the above experiment, the inflow (source) and outflow (drain) boundaries are prescribed by subsonic boundary conditions. For a subsonic inflow boundary, the quantity nu (current) needs to be specified. Since the current at the boundary is an unknown quantity, only the concentration is specified. An alternative, the second set of boundary conditions discussed in Section 6 can be imposed for subsonic inflow. More interestingly, at the outflow boundary, the results that we have observed indicate that the flow is not entirely subsonic. Towards the edge of the drain contact, a number (about 3 or 4)

of mesh nodes on the outflow boundary exhibit supersonic flow.

In the second experiment, we simulated the same device with the outflow boundary specified to be supersonic. Two possible sets of boundary conditions can be specified for supersonic outflow boundaries. In Figures 4 and 5 we show the electron concentration and temperature when only the temperature is prescribed at the drain contact. In Figures 6 and 7 we show the electron concentration and temperature when $\frac{\partial T}{\partial n} = 0$ is specified on the drain contact. As shown in these figures, different global solutions could be obtained based on the boundary conditions specified. The complication of this example is due to the subsonic outflow except for a few mesh nodes at the edge of the drain contact. Although not being implemented, this situation can be handled by implementing point based boundary conditions where each mesh node is checked for subsonic/supersonic outflow before boundary conditions are specified.

The last experiment demonstrates the case where the outflow boundary is assumed to be subsonic but is overspecified. Figures 8 shows the electron temperature when $n, u, T, \frac{\partial T}{\partial n}$ are specified for the drain contact. A small overshoot can be observed in the temperature profile near the drain end. This result suggests that overspecification of boundary conditions should be avoided where possible.

9 Conclusion

In this paper we have analyzed the boundary conditions for the well-posedness of the hydrodynamic equations for semiconductor devices. We have shown that the specification of boundary conditions for HD equations is different from the Navier-Stokes equations. Furthermore, we have shown that the boundary conditions for the outflow boundaries are different from those of Euler equations. We have also shown that the heat conduction term plays an important role in deriving the number of independent conditions and cannot be neglected in deriving well-posed boundary conditions.

Two different symmetrization approaches are discussed for the HD equations. The two symmetrization approaches lead to similar results on the requirement in the selection of proper boundary conditions. Several sets of boundary conditions are presented for the inflow and outflow boundaries. We observe that some commonly employed boundary conditions do not give well-posedness to the HD equations. Boundary conditions for subsonic inflow require further investigation for devices with high level injection.

The analysis presented in this paper assumes that the Poisson and the HD equations are solved using a decoupled staggered numerical strategy [8] (similar to the well-known Gummel scheme [10]). If a coupled scheme (in which Poisson and HD equations are solved as a single system) is employed to solve the semiconductor device equations, the boundary conditions discussed in this paper may not carry over to such cases but should give some insight to the problem. An analysis of coupled semiconductor equations is beyond the scope of this paper.

10 Acknowledgments

This research is sponsored by ARPA through contract # DAAL 03-91-C-0043. The authors would like to thank the helpful discussions with Drs. Ke-Chih Wu and Arthur Raefsky.

References

- 1 C. Gardner, J. W. Jerome and D. J. Rose, "Numerical methods for the hydrodynamic device model: subsonic flow", IEEE Trans. CAD, Vol. 8, pp. 501-507, 1989.
- 2 K. Blotekjaer, "Transport equations for electrons in two-valley semiconductors", IEEE Trans. Elec. Dev., Vol. ED-17, pp. 38-47, 1970.
- 3 E. Thomann and F. Odeh, "On the well-posedness of the two-dimensional hydrodynamic model for semiconductor devices", COMPEL, Vol. 9, pp. 45-57, 1990.
- 4 S. W. Bova and G. F. Carey, "An analysis of the time-dependent hydrodynamic device equations", Proc. Intl. Workshop Comp. Elec., Univ. of Illinois, pp. 91-94, May 28-29, 1992.
- 5 J. Strikwerda, "Initial boundary value problems for incompletely parabolic systems", Ph. D. thesis, Dept. of Math., Stanford Univ., Stanford, CA, 1976.
- 6 B. Gustafson and A. Sundstrom, "Incompletely parabolic problems in fluid dynamics", SIAM J. Appl. Math., Vol. 35, pp. 343-357, 1978.
- 7 J. Olinger and A. Sundstrom, "Theoretical and practical aspects of some initial boundary value problems in fluid dynamics", SIAM J. Appl. Math., Vol. 35, pp. 419-446, 1978.
- 8 N. R. Aluru, A. Raefsky, P. M. Pinsky, K. H. Law, R. J. G. Goossens and R. W. Dutton, "A finite element formulation for the hydrodynamic semiconductor device equations", Comp. Meth. Appl. Mech. Engg., Vol. 107, pp. 269-298, 1993.
- 9 A. Harten, "On the symmetric form of systems of conservation laws with entropy", J. Comp. Phys., Vol. 49, pp. 151-164, 1983.
- 10 H. K. Gummel, "A self-consistent iterative scheme for one-dimensional steady state transistor calculations", IEEE Trans. Elec. Dev., Vol. ED-11, pp. 455-465, 1964.
- 11 N. R. Aluru, K. H. Law, P. M. Pinsky, A. Raefsky, R. J. G. Goossens and R. W. Dutton, "Space-time Galerkin/least-squares finite element formulation for the hydrodynamic device equations", IEICE Trans. Electron., Vol. E77-C, No. 2, pp. 227-235, 1994.
- 12 M. Sever, "Symmetric forms of energy-momentum transport models", IMA Vol. 59, pp. 365-376, 1994.
- 13 T. J. R. Hughes, L. P. Franca and G. M. Hulbert, "A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective-diffusive systems", Comp. Meth. Appl. Mech. Engg., Vol. 58, pp. 173-189, 1986.

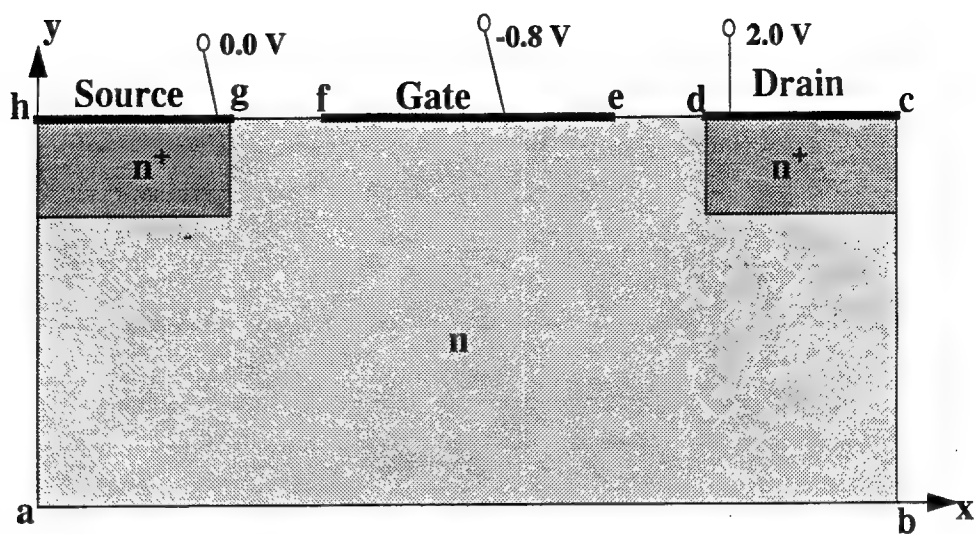


Figure 1 A two-dimensional MESFET device

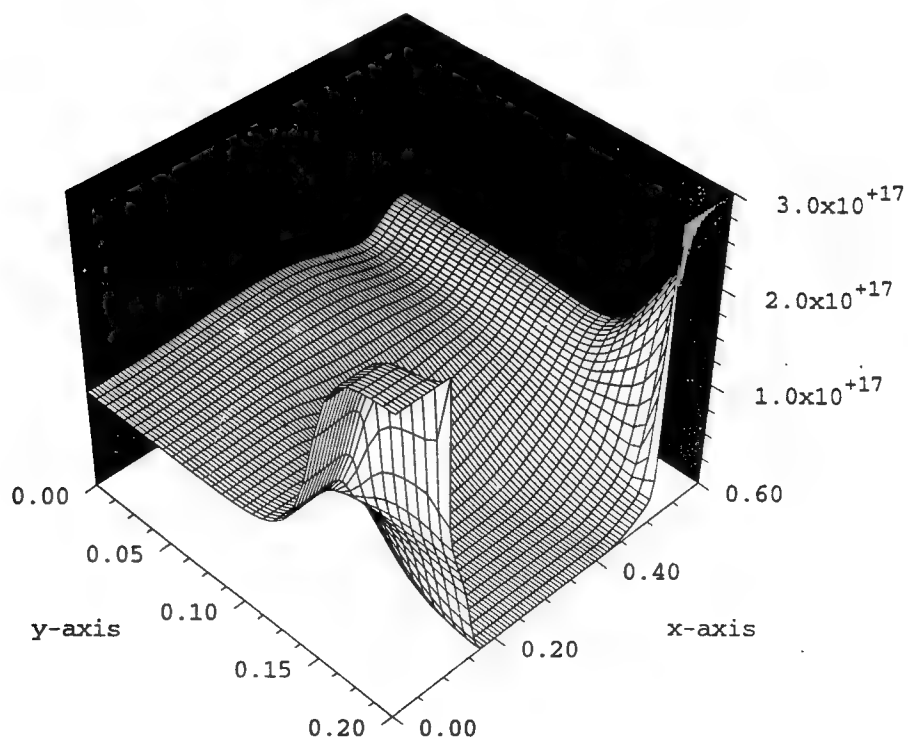


Figure 2 Electron concentration for subsonic outflow

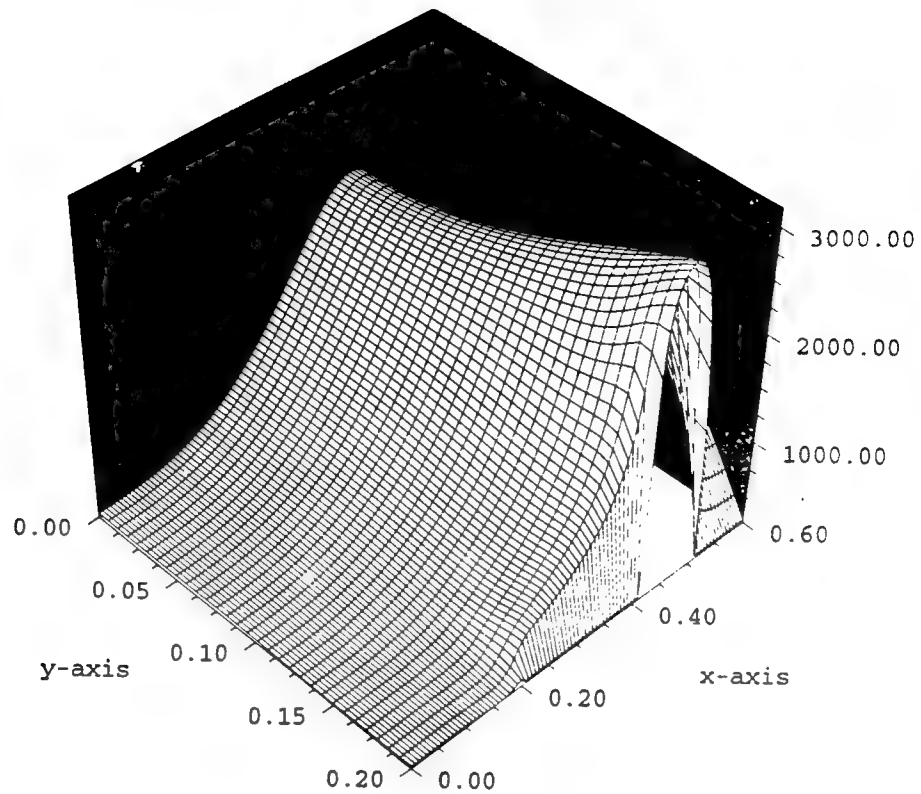


Figure 3 Electron temperature for subsonic outflow

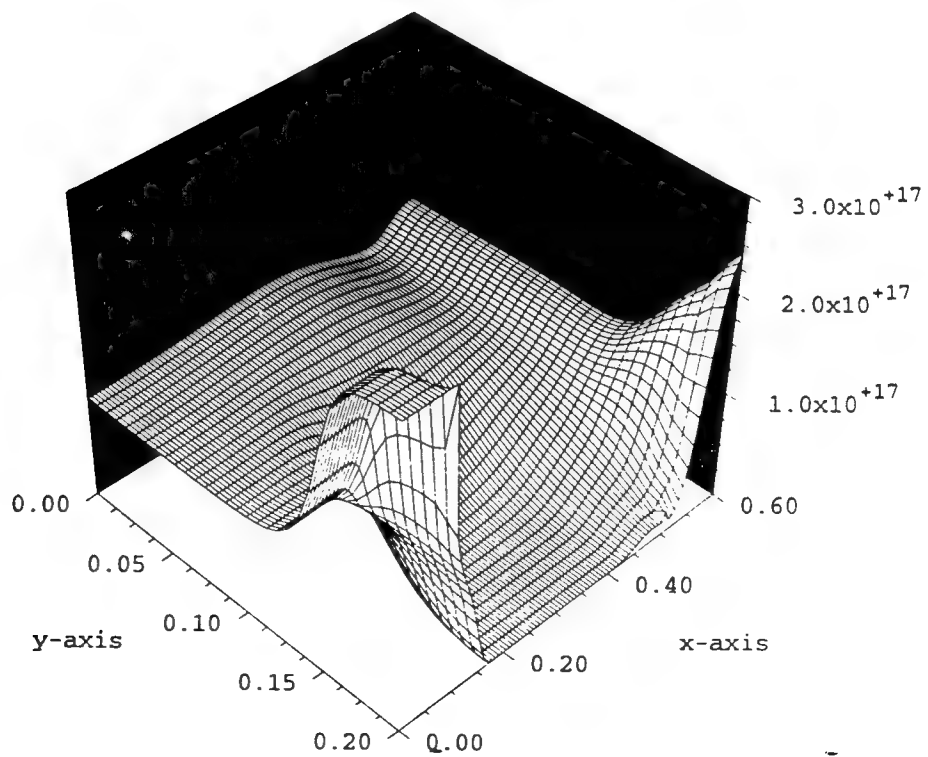


Figure 4 Electron concentration for supersonic outflow with T specified on drain

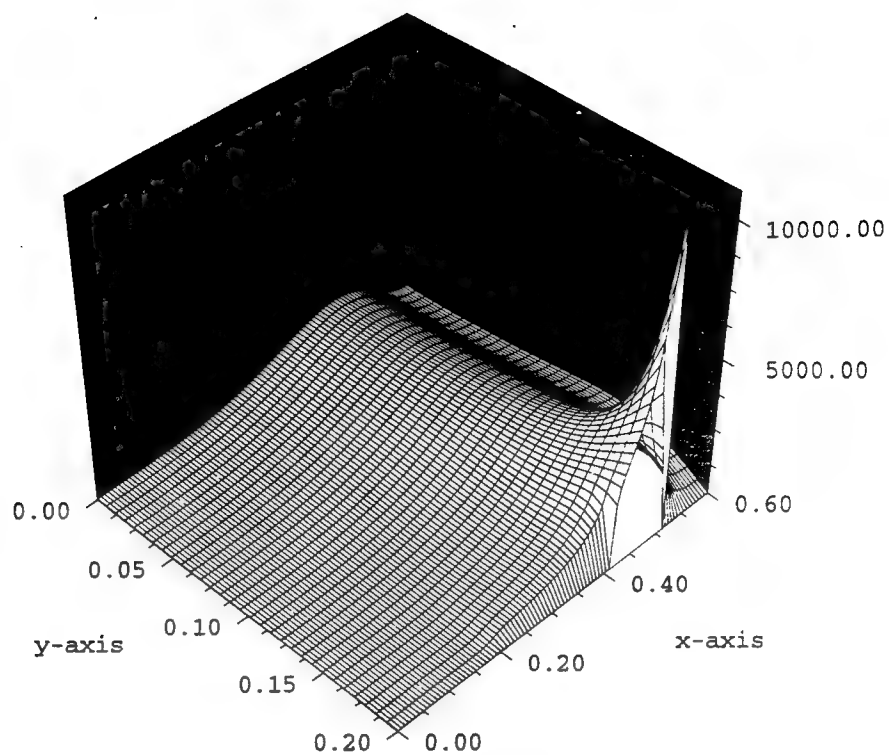


Figure 5 Electron temperature for supersonic outflow with T specified on drain

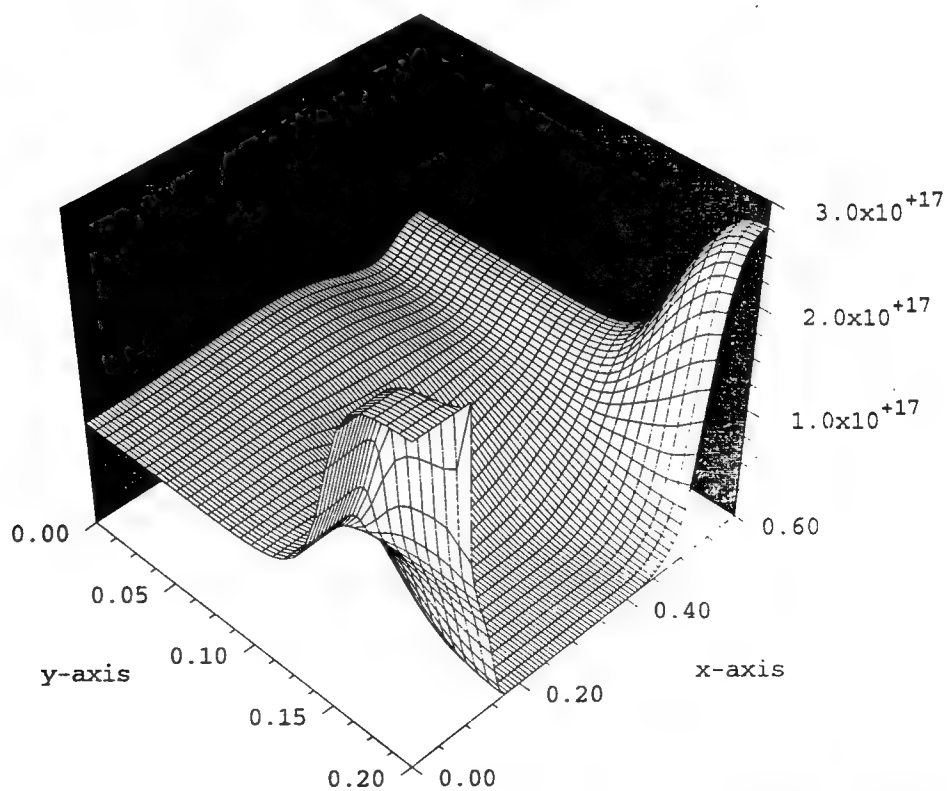


Figure 6 Electron concentration for supersonic outflow with $\frac{\partial T}{\partial n}$ specified on drain

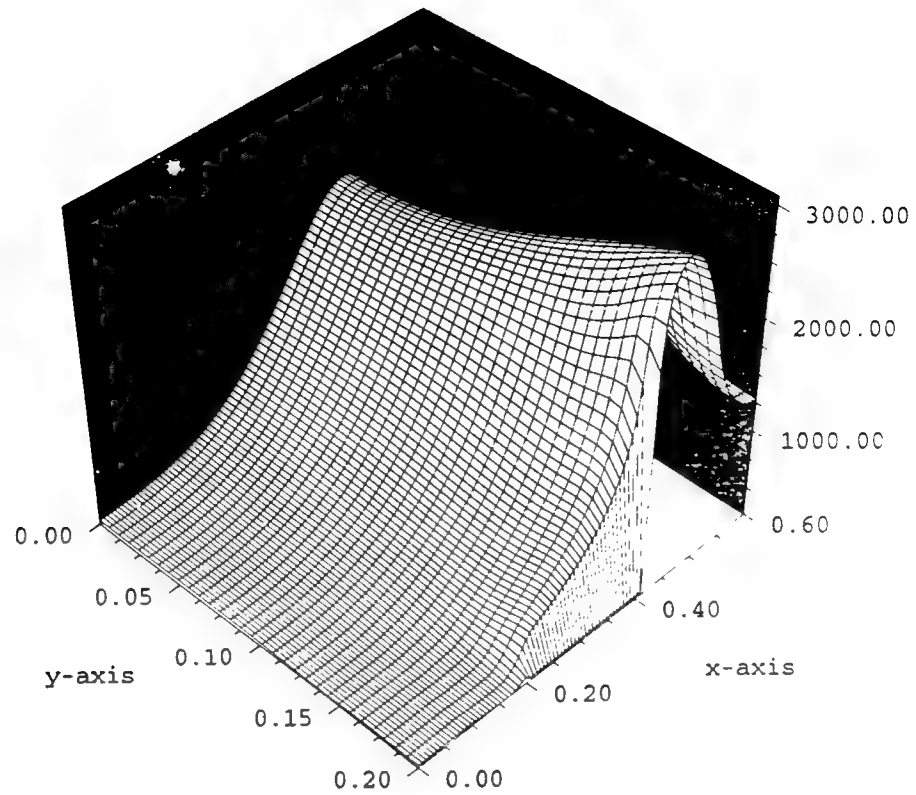


Figure 7 Electron temperature for supersonic outflow with $\frac{\partial T}{\partial n}$ specified on drain

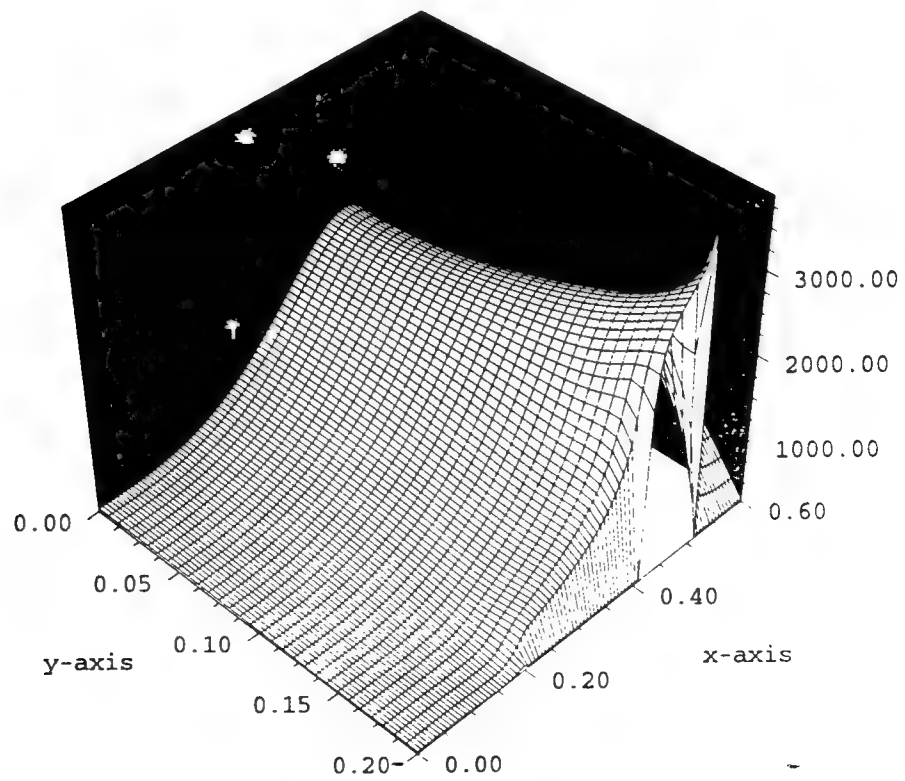


Figure 8 Electron temperature for subsonic outflow with overspecified data on drain

FIESTA-HD : A Parallel Finite Element Program for Hydrodynamic Device Simulation¹

Narayana R. Aluru, Kincho H. Law, Arthur Raefsky and Robert W. Dutton
231-F, Applied Electronics Laboratory, Stanford University, Stanford, CA 94305-4020.

Extended Abstract

Numerical simulation of the hydrodynamic semiconductor device model involves the solution of a coupled system of partial differential equations; namely, the Poisson equation for the electric field and the hydrodynamic (HD) equations for the electron and hole carriers. Motivated by the success of the Galerkin/Least Squares (GLS) finite element method in computational fluid dynamics and the resemblance of the HD equations to the Euler and the Navier-Stokes fluid equations, we extend the GLS method to account for the strong nonlinear source terms and apply the method to the HD equations for semiconductor devices. The complexity of the coupled system for the HD model demands enormous computational time. A parallel finite element device simulation program, FIESTA-HD, has been developed and run on various distributed memory parallel computers. In the following, we introduce briefly the hydrodynamic device model, discuss the finite element formulations employed and describe the parallel implementation model.

Hydrodynamic Model for Semiconductor Device Simulation

Semiconductor device simulations employing the hydrodynamic model involve the solution of the coupled nonlinear system of Poisson equation for the description of electrostatic potential and electric field, and the hydrodynamic conservation laws for the description of the carrier concentration, velocity and temperature. Derived from the Maxwell's equations, the Poisson equation for computing the electrostatic potential and the electric field can be summarized as:

$$\nabla \cdot (\theta \nabla \psi) = \varepsilon(c_n - c_p - N_D^+ + N_A^-) \quad \text{and} \quad \mathbf{E} = -\nabla \psi$$

where ε , ψ , θ and \mathbf{E} are the charge, the permittivity, the electrostatic potential and the electric field, respectively; c_n , c_p , N_D^+ and N_A^- are the concentrations of electrons, holes, ionized donors and ionized acceptors, respectively. The subscripts n and p denote, respectively, the electron carrier and the hole carrier.

The electron and hole hydrodynamic equations can be derived from the first three moments of the Boltzman Transport equation (BTE):

$$\begin{aligned} \frac{\partial c_\alpha}{\partial t} + \nabla \cdot (c_\alpha \mathbf{u}_\alpha) &= \left[\frac{\partial c_\alpha}{\partial t} \right]_{col} \\ \frac{\partial \mathbf{p}_\alpha}{\partial t} + \mathbf{u}_\alpha \cdot (\nabla \mathbf{p}_\alpha) + (\mathbf{p}_\alpha \cdot \nabla) \mathbf{u}_\alpha &= (-1)^j \varepsilon c_\alpha \mathbf{E} - \nabla (c_\alpha k_b T_\alpha) + \left[\frac{\partial \mathbf{p}_\alpha}{\partial t} \right]_{col} \\ \frac{\partial w_\alpha}{\partial t} + \nabla \cdot (\mathbf{u}_\alpha w_\alpha) &= (-1)^j \varepsilon c_\alpha (\mathbf{u}_\alpha \cdot \mathbf{E}) - \nabla (\mathbf{u}_\alpha c_\alpha k_b T_\alpha) - \nabla \cdot \mathbf{q}_\alpha + \left[\frac{\partial w_\alpha}{\partial t} \right]_{col} \end{aligned}$$

where \mathbf{u}_α , \mathbf{p}_α , T_α , w_α and \mathbf{q}_α are the velocity vector, momentum density vector, temperature, energy density and heat flux vector of the carrier α . (For electron, $\alpha = n$ and $j = 1$; for holes, $\alpha = p$ and $j = 2$.) The terms $[]_{col}$ represent the rate of change in the particle concentration, momentum and energy due to the collision of carriers; the collision terms can be approximated by their respective relaxation times and the expressions can be found in Ref. [1]. The following constitutive relations are appended to the above equations to facilitate the solution:

$$\mathbf{p}_\alpha = m_\alpha c_\alpha \mathbf{u}_\alpha \quad \text{and} \quad w_\alpha = \frac{3}{2} c_\alpha k_b T_\alpha + \frac{1}{2} m_\alpha c_\alpha |\mathbf{u}_\alpha|^2$$

where m_α is the mass density of the carrier and k_b is the Boltzman constant.

Similar to the Euler and Navier-Stokes fluid equations, the HD equations can be physically interpreted as the conservation of particle, momentum and energy. However, the HD equations are not

¹Submitted to Parallel CFD'95, California Institute of Technology, Pasadena, CA, June 26-28, 1995.

identical to either the Euler or the Navier-Stokes equations. While the HD equations do not contain the viscous terms, they are not the same as the Euler equations because of the presence of the heat conduction term in the energy equations. Furthermore, the highly nonlinear source terms in the HD model are absent in the fluid models. It can be shown that the HD system resembles the flow of an ideal compressible fluid given by the Euler equations, in the presence of electric field and with the addition of a heat conduction term and the highly nonlinear source terms.

Finite Element Formulation

For the elliptic Poisson equation, a standard Galerkin finite element method has been employed for the numerical solution. However, the standard Galerkin finite element method is known to exhibit spurious oscillations for the advective-diffusive type equations like the HD equations when the physical diffusion present in the system is small. In this work, we employ the Galerkin/Least-Squares (GLS) method [3] and extend it to account for the strong nonlinear source terms of the HD device equations. The temporal behavior of the HD equations is discretized using a discontinuous Galerkin method in time [4]. The basic formulation of the space-time GLS discretization scheme can be summarized as follows:

1. A least-squares term of a residual type is introduced to the weak form of the given partial differential equation so that the numerical stability of the system is enhanced. Furthermore, a discontinuity-capturing term is added to overcome the undershoot and overshoot phenomena. The least-squares and discontinuity capturing terms vanish when the exact solution is substituted to the weak form.
2. The trial and test functions are approximated by linear basis functions.
3. The nonlinear system is solved using a Newton iterative scheme by linearizing the nonlinear equations with respect to the unknown trial solution.

A comprehensive discussion on the development of the finite element space-time GLS formulation for the HD semiconductor device equations is given in Ref. [1].

A staggered scheme is applied to solve the coupled systems. The Poisson equation is first solved for the electrostatic potential and the electric field. The computed electric field values are used in the HD equations to solve for the concentrations, velocities and temperature. The concentrations obtained from the HD equations result in a new source term to the Poisson equation. This staggered procedure of alternatively solving the Poisson and HD equations is repeated until both the equations are solved to a desirable tolerance.

Parallel Computational Model

The single-program-multiple-data (SPMD) paradigm has emerged as a standard model to create parallel programs for engineering applications on distributed memory parallel computers [2]. In this approach, problems are decomposed using some well known domain decomposition techniques. Each processor of the parallel machine solves a partitioned domain. Data communication between domain partitions are performed among processors through message passing.

For a large scale engineering software, besides optimizing the parallel kernels for linear algebraic and/or matrix computations, attention must be paid to the overall program structure and the data flow among the program modules. A typical finite element program consists of the following tasks: pre-processing, element generation, matrix formation, solution of a system of linear equations and post-processing. The pre-processor supports problem definition, grid generation, I/O and other file management functions. Generally, the pre-processing routines take negligible time and are inherently serial. The parallelization is thus concentrated on the numerical PDE solvers. In FIESTA-HD, the linear equation solvers currently employed are GMRES for solving the non-symmetric linear equations of the HD systems and conjugate gradient for the symmetric linear equation for the Poisson system. For a finite element program with iterative solvers, the parallel communication is limited primarily to the linear solver. Special care, however, is needed to set up the data structures required by each

processor and to ensure proper data flow between the pre-processor and the parallel PDE solvers. The parallel program organization of FIESTA-HD is depicted as shown in Fig. 1.

Initial development of the parallel FIESTA-HD program took place on a 32-node Intel iPSC/860 computer. The code has since been ported to the Intel Touchstone Delta and the IBM SP1 computers. For the Intel-based implementation, a front end workstation is used for the pre-processing tasks. For the IBM SP1 parallel computer, the pre-processor resides on a master node (which also serves as a slave processor for the parallel PDE solvers) and a more efficient model is implemented, taking advantage of the memory available on the SP1. The porting of the code from the iPSC/860 to the Delta and to the SP1 takes less than a week. For each case, majority of the work has been to re-structure the pre-processing module.

To demonstrate the utility of FIESTA-HD, we have run simulations using increasingly large and complex realistic device structures on the parallel computers and on an IBM RS/6000 Model 530 workstation. The results are summarized as shown in Fig. 2. The results clearly show the portability and scalability of the simulator on various parallel computers. As grids scaled to modest and large sizes, the parallel codes perform significantly better than the workstation version. We routinely achieve more than an order-of-magnitude reduction in execution time. Moreover, using these parallel machines, we have been able to solve very large device structures for which a serial solution could not be obtained due to resource constraints.

Summary and Discussion

In this note, we have briefly discussed the hydrodynamic model for semiconductor device simulation and the resemblance of the HD device equations with the Euler and Navier-Stokes fluid equations. A space-time Galerkin/Least-Squares finite element method is proposed for the solution of the HD equations. A SPMD programming model is used in the parallel implementation of the device simulator, FIESTA-HD. Our experience has clearly demonstrated the portability of FIESTA-HD on distributed memory parallel computers. Other features, such as the lattice thermal diffusion equation describing the variation of the lattice temperature in the semiconductor device, are currently being incorporated. Taking advantage of the advances in parallel computers with stable numerical schemes, we are able to perform simulations with more complex and realistic device models.

Acknowledgement

This research was sponsored by ARPA, contract No. DAAL 03-91-C-0043.

References

- [1] N.R. Aluru, A. Raefsky, P.M. Pinsky, K.H. Law, R.J.G. Goossens and R.W. Dutton, "A Finite Element Formulation for the Hydrodynamic Semiconductor Device Equations," *Computer Methods in Applied Mechanics and Engineering*, 107:269-298, 1993.
- [2] B.F. Herndon, N.R. Aluru, A. Raefsky, R.J.G. Goossens, K.H. Law and R.W. Dutton, "A Methodology for Parallelizing PDE Solvers: Application to Semiconductor Device Simulation," Seventh SIAM Conference on Parallel Computing, San Francisco, CA, 1995.
- [3] F. Shakib, *Finite Element Analysis of the Compressible Euler and Navier-Stokes Equations*, Ph.D. Thesis, Department of Mechanical Engineering, Stanford University, Nov., 1988.
- [4] C. Johnson, U. Navert and J. Pitkaranta, "Finite Element Methods for Linear Hyperbolic Problems," *Computer Methods in Applied Mechanics and Engineering*, 45:285-312, 1984.

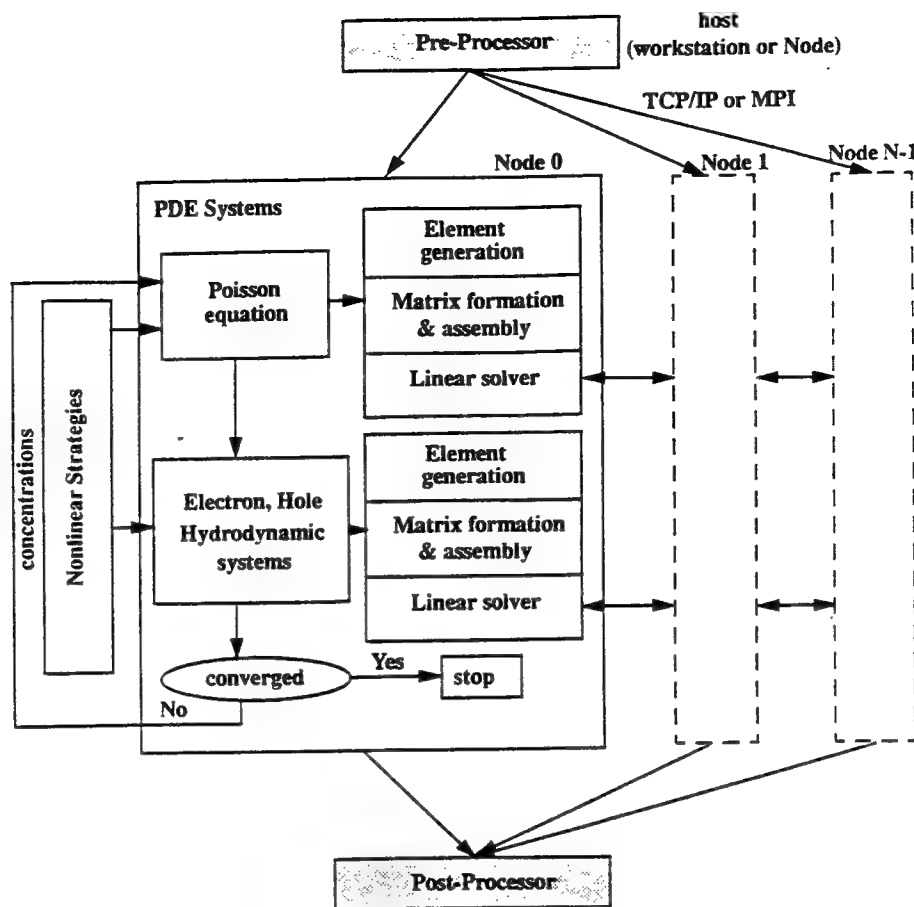


Figure 1: Program Organization of Parallel HD Device Simulator, FIESTA-HD

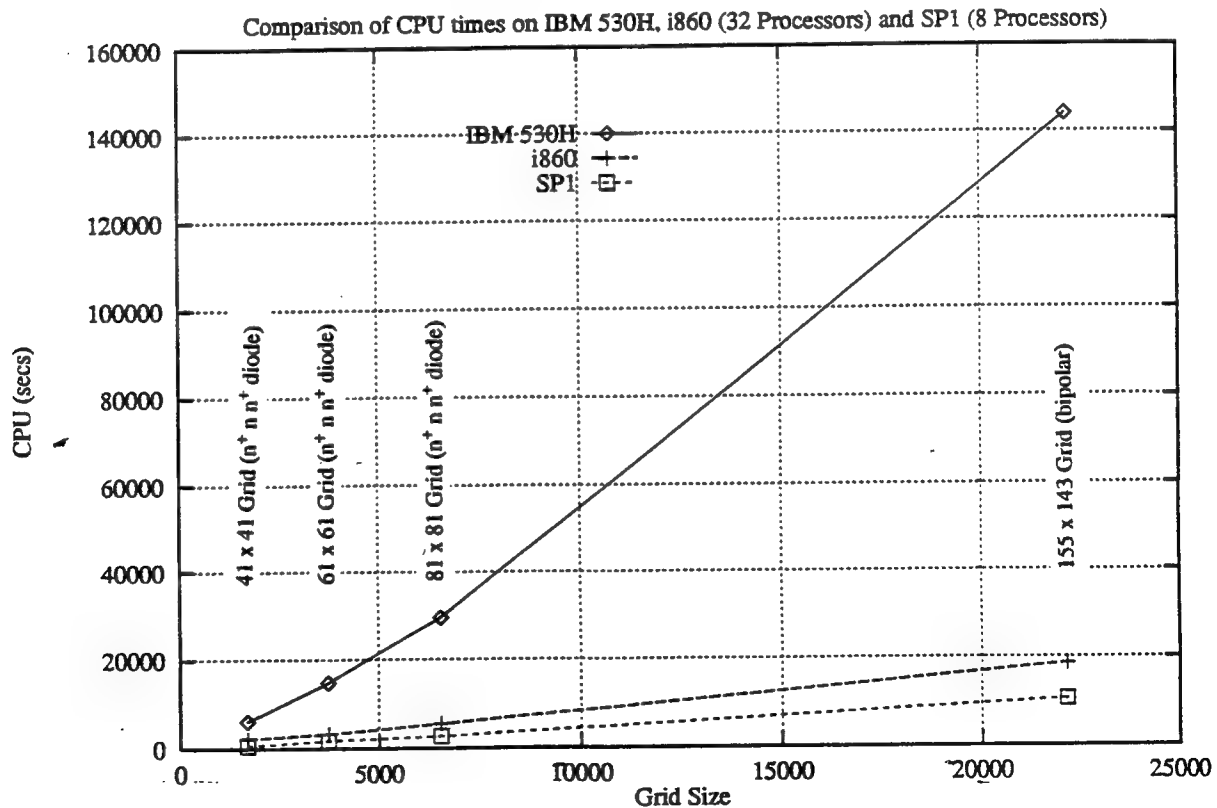


Figure 2: Comparison of Execution Times of FIESTA-HD on Parallel Computers

Parallel PISCES

Robert Lucas and Tom Blank

AEL 231D Integrated Circuits Laboratory
Stanford University, Stanford, CA 94305
(415) 723-1482

Abstract

This paper presents a parallel implementation of Stanford's PISCES¹, a two dimensional device analysis program. It offers a practical solution to the critical computational bottleneck now facing IC device designers. A nested dissection of the problem grid is used to preserve spatial locality in the distribution of the problem amongst the processors. Both assembly and decomposition of the sparse matrices used in the Newton iterations is parallelized. The matrix assembly operation proceeds concurrently without communication, yielding near perfect speedup. The matrix is decomposed and the forward and back solutions performed using the new distributed multifrontal algorithm². Ten fold speedups of the Newton iterations are shown to be feasible on a sixteen processor hypercube.

1. Introduction

Numerical simulation of integrated circuit device behavior using Stanford's PISCES program involves discretizing partial differential equations that model the device's behavior and then solving the resulting algebraic equations by Newton's method. This requires repeated solution of large sparse systems of linear equations. Tight coupling of these equations mandates the use of sparse LU decomposition³. The repeated assembly and decomposition of these large matrices can account for as much as 95% of the run time of a two dimensional numerical device simulation⁴. These problems are outgrowing the traditional von Neumann style computers available to most device designers.

The advent of VLSI has made it possible to solve computationally intensive problems such as device simulation with large ensembles of inexpensive processors. These range from systolic arrays⁵ which implement specific algorithms to large multiprocessors containing hundreds of general purpose CPUs. A distributed memory, message passing hypercube was chosen as the environment in which a parallel implementation of the PISCES device simulator was performed. The message passing hypercube architecture requires only $\log_2 N$ communication ports per processor to connect N processors. Each processor is linked to every other processor whose binary identifier differs by only one digit. These machines lend themselves to massively parallel implementations and there are currently four such machines commercially available.

The basic Gaussian Elimination algorithm is inherently parallel and there has been a great deal of research into implementing sparse LU decomposition on multiprocessors⁶. A theoretical lower bound on the number of operations required was derived by Wing⁷. Implementations of sparse matrix factorization for shared memory machines have been studied by

iPSCTM is a trademark of Intel Corporation

Duff⁸, Alghband⁹, and Jacob¹⁰. Implementations for the hypercube have been reported by Geist¹¹, Clinard¹², and Lucas². Of these, only the work of Clinard involved the parallel assembly of a complete application problem.

This paper presents a parallel implementation of Stanford's PISCES two dimensional device simulator. The solution by Newton's method of the partial differential equations that model device behavior is performed on an Intel iPSC/MXTM hypercube. Each processor (node) in the hypercube contains an Intel 80286, a 80287 floating point co-processor, and 4.5 megabytes of memory. Communication with the hypercube is through its 80286 based Intel 310 host processor. The paper is organized as follows: Part two reviews PISCES and motivates the division of the problem into portions that run sequentially on the host, and those which run in parallel on the hypercube. Part three discusses the problem distribution among the processors and includes matrix ordering, symbolic decomposition, and the concurrent assembly and solution of the sparse system of equations. In part four an analysis of the performance of the parallel simulator is presented. Finally, conclusions are drawn and implications of this work are suggested.

2. Review of PISCES

The bulk behavior of semiconductor devices is modeled by three partial differential equations (PDEs). Poisson's equation governs the electrostatic potential (ψ) and the electron and hole continuity equations govern the carrier concentrations (n and p). For reference, the equations are listed below:

$$e\nabla^2\psi = -q(p - n + N_D^+ - N_A^-) - \rho_F \quad (2.1)$$

$$\frac{\partial n}{\partial t} = \frac{1}{q} \nabla \cdot J_n - U_n \quad (2.2)$$

$$\frac{\partial p}{\partial t} = -\frac{1}{q} \nabla \cdot J_p - U_p \quad (2.3)$$

where N_A^- and N_D^+ are the ionized impurity densities, ρ_F is a fixed charge density that may be present in insulating materials, J_n and J_p are the electron and hole current densities, and finally, U_n and U_p are the electron and hole recombination rates. Details of the discretization of these equations on a simulation grid can be found in Craig Price's thesis¹³ and in the PISCES technical report, therefore, they will not be repeated. When discretized, these three PDEs form a coupled set of non-linear algebraic equations. There is no direct method to solve them in one step. Consequently, solutions are obtained by using either Gummel's or Newton's method of non-linear iteration. Since Gummel's method converges slowly when the device being simulated is in high injection, Newton's method is preferred.

In Newton's method, the equations are expressed as follows:

$$G_\psi(\psi, n, p) = 0$$

$$G_n(\psi, n, p) = 0$$

$$G_p(\psi, n, p) = 0$$

Given an initial guess for the values of ψ , n , and p at each node, a new update ($\Delta\psi$, Δn , Δp) is computed by solving the linear system

$$\begin{bmatrix} \frac{\partial G_n}{\partial \psi} & \frac{\partial G_n}{\partial n} & \frac{\partial G_n}{\partial p} \\ \frac{\partial G_p}{\partial \psi} & \frac{\partial G_p}{\partial n} & \frac{\partial G_p}{\partial p} \\ \frac{\partial G_v}{\partial \psi} & \frac{\partial G_v}{\partial n} & \frac{\partial G_v}{\partial p} \end{bmatrix} \begin{bmatrix} \Delta\psi \\ \Delta n \\ \Delta p \end{bmatrix} = - \begin{bmatrix} G_n \\ G_p \\ G_v \end{bmatrix}$$

The updates are added to ψ , n , and p and the process repeated until it converges to a stable solution.

The matrix is assembled from a two dimensional simulation grid that describes the physical structure of the device. Figure 1 represents a small diode discretized on a 15 by 15 grid. The equations are discretized using the box method¹⁴ such that each equation is integrated over a small polygon enclosing a node of the simulation grid. The integration equates the flux into the polygon with the sources and sinks inside it. The integrals are performed independently, one triangle at a time. Therefore, assembly of the sparse matrix and the right hand side vector (RHS) can be performed concurrently, each processor assembling a unique subset of the triangular elements.

Once the Jacobian has been assembled, it is decomposed into its lower and upper triangular factors. The updates to ψ , n , and p are then computed by forward elimination and back substitution. To minimize the number of arithmetic operations on a sequential processor, the matrix is reordered using the minimum degree algorithm. Convergence is attained and the Newton iteration terminated when the magnitude of either the updates or the RHS have fallen below specified tolerances.

The repeated assembly and decomposition of these large sparse matrices is the computational bottleneck of PISCES. This is demonstrated with a simulation of the diode depicted in Figure 1. Figure 2 contains the input deck that defines the problem, a simulation of the forward characteristic of a diode at three successive bias points. The first six MESH cards define the simulation grid (i.e. Figure 1). The next five cards define the physical structure of the device and its contacts. The next three specify parameters of the simulation. Finally, the SOLVE cards instruct PISCES to actually perform the simulation. Table 1 shows the total CPU time as well as the cumulative time spent in the Newton iterations and its key assembly, factorization and triangular solution subroutines. These CPU times are also represented as percentages of the total run time. The simulation was run on the hypercube's host processor and illustrates how the Newton iterations dominate the run-time of PISCES. The host's XENIXTM 286 R3.4 operating system limits the run time size of a program to under 3 megabytes. Therefore, the data structures used in PISCES had to be reduced to port the code to the host and it can only simulate small devices.

	Total Time	Newton Routines			
		Assemble	Factor	Solve	Other
Run Time	1165	421	568	80	25
Percentages	100%	36%	49%	7%	2%

Table 1
Time (sec.) and percentage of total time that PISCES spent executing key routines in the sample program in Figure 2

Table 1 shows that even on the small diode problem PISCES spent 94% of its time in the Newton iterations. The remaining time was spent parsing the command file, generating the mesh, reordering the matrix, and performing the symbolic decomposition. As the size of the simulation grid grows, the run-time of the sparse matrix factorization grows superlinearly. Therefore, on the larger problems commonly simulated, the Newton iterations, specifically the sparse matrix factorizations, dominate the throughput. This is the motivation for a parallel implementation of Newton's method.

XENIXTM is a trademark of Microsoft Corporation

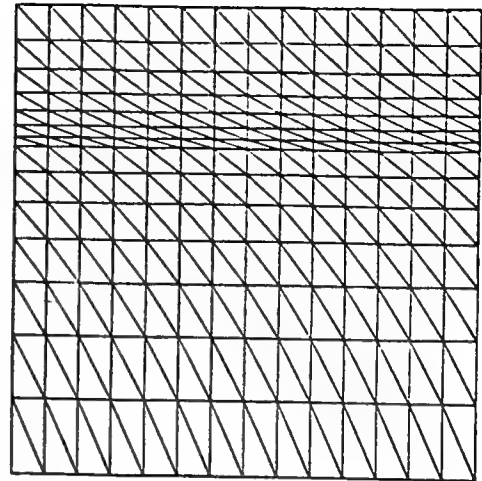


Figure 1
15 by 15 simulation grid for a pn junction diode

```
title square pn diode
mesh rect nx=15 ny=15
x.mesh location=0.0 node=1 ratio=1
x.mesh location=1.0 node=15 ratio=1
y.mesh location=0.0 node=1 ratio=1
y.mesh location=0.3 node=8 ratio=0.8
y.mesh location=1.0 node=15 ratio=1.2
region num=1 silicon ix.lo=1 ix.hi=15 iy.lo=1 iy.hi=15
elec num=1 ix.lo=1 ix.hi=15 iy.lo=1 iy.hi=1
elec num=2 ix.lo=1 ix.hi=15 iy.lo=15 iy.hi=15
doping reg=1 n.type conc=1e15 uniform
doping reg=1 p.type conc=1e19 gauss x.l=0 x.r=1
+ y.top=0 y.bot=0 junc=0.3
symb newton cube can=2
method rhsnorm xnorm autonr
models temp=300 srh auger conmob fldmob
solve init
solve vstep=0.1 nsteps=3 elect=1
end
```

Figure 2
PISCES input deck that defines the simulation of the diode in Figure 1

3. Parallel PISCES

Execution of PISCES begins with the parsing of the user's input deck. This is an I/O intensive operation that requires input of a file that defines the correct user syntax, input of the user's input deck, and then output of the parsed commands to a temporary file. File I/O from the hypercube is implemented by sending a message to a background process on the host which then performs the requested operation and returns the result via another message. This process is substantially slower than file I/O on the host. Therefore, the parsing of the user's input is implemented on the host processor. PISCES is informed of the existence of the hypercube when it encounters a SYMBOL card in the user's input deck in which the CUBE flag has been set. The state of the program is then transferred to the hypercube upon receipt of a subsequent SOLVE card. This is accomplished by transmitting, in their entirety, all of the FORTRAN COMMON blocks that define the permanent storage visible to the subroutines called by the SOLVE card. While this involves the transfer of a tremendous volume of

memory (currently 524,520 bytes), it allows the routines that run on the host to be ported to the parallel processor with a bare minimum of effort. PISCES has been treated as a "dusty deck" and only a small number of key subroutines have been modified. This permits the same routines that run on the hypercube to also run on the host processor where there are superior debugging facilities.

The state transfer is initiated by the host which transmits the COMMON blocks to Node 0 of the hypercube. It takes a total of 23 seconds to transmit the common blocks from the host to Node 0. These are long messages and as such are as efficient as the communication primitives permit. Upon receipt of each COMMON block, Node 0 uses a spanning tree¹⁷ to distribute the data to the remaining processors in the hypercube. The spanning tree allows data to be distributed to $N-1$ processors in the time it takes to transmit $\log_2 N$ messages. Each increase in the dimension of the hypercube adds approximately 8 seconds to the time required to initialize it. Thus it takes 31 seconds to initialize a one dimensional hypercube (two processors) and 54 seconds to initialize a four dimensional hypercube (16 processors).

PISCES execution on the hypercube begins with a reordering of the grid and then the symbolic decomposition of the sparse matrix. These functions are normally performed after receipt of the SYMBOL card. In Parallel PISCES, they are deferred until the execution of the SOLVE card to reduce message traffic between the host and the hypercube. Each processor receives a complete description of the entire grid and isolates, via an incomplete, nested dissection¹⁸, its own block in the grid. This prevents the need to initialize each processor with a unique message that defines its subset of the problem. At each stage of the dissection process, a separator is found which divides the grid into two blocks. This process is recursively applied to each block until a block has been isolated for each processor. Blocks are allocated to processors based upon their physical location in the hypercube and each processor independently identifies the separators that isolate its block. Once the separators in the grid that isolate the blocks have been chosen, each processor is free to reorder the nodes in its block independently. No messages need be exchanged between any processors. Figure 3 is an example of the distribution of a 15 by 15 grid over 16 processors.

The symbolic decomposition creates a template that will later allow exploitation of the non-zero structure of the sparse matrix such that the number of floating point arithmetic operations needed to factor the matrix is limited. Each processor treats its block as an entirely local sparse problem. Details of the sparse structure of the blocks are shielded from the other processors

by the separators and no messages need be exchanged to decompose them symbolically. The separators are treated as dense sub-problems. Since every processor is fully cognizant of the details of the separators that isolated its block, symbolic decomposition of the separators also requires that no messages be exchanged.

In part two of this paper it was mentioned that the assembly of the matrix representing the sparse system of equations proceeds one triangle at a time and that these triangles can be processed independently of one another. The natural way to exploit this concurrency on the hypercube is to allow each processor to assemble the triangles that reside in its local block of the simulation grid. Implementing this required the addition of only five lines of FORTRAN code to the PISCES ASSMBL subroutine. Rather than loop through all of the triangles in the grid, the ASSMBL subroutine of Parallel PISCES need only loop through those identified as being in its local block. No processor computes values for any locations in the matrix that do not correspond to vertices of the grid that are in its block or in the separators that bound the block. Therefore, no messages are required to store the assembled values.

Sparse matrix factorization is accomplished using the distributed multifrontal (DMF) algorithm. The blocks are factored independently and updates computed for the separators are accumulated locally. This defers the exchange of messages between processors until there remain only the separators to factor. The separators are then factored cooperatively. They correspond to small dense problems that can be factored efficiently even in an environment where the throughput of the arithmetic processors is two orders of magnitude greater than that of the communication channels¹⁹. The number of messages exchanged is limited to a function of the lengths of the separators and thus very sparse matrices can be factored efficiently on a message passing multiprocessor.

Following the triangular solutions, a spanning tree is used to collect and then distribute all of $\Delta\psi$, Δn , and Δp to each processor. Every processor then performs the tasks of updating the solution (ψ , n , and p) and checking for convergence over the entire grid. Like the matrix assembly phase, this could be parallelized by restricting each processor to its local block of the grid. Global information such as the norm of the solution vector could then be efficiently collected using a spanning tree. However, this portion of each Newton loop runs very quickly (.25 sec. in the example in Table 1) and thus its serial execution imposes only a minor reduction in the overall efficiency of Parallel PISCES. In the spirit of parallelizing PISCES in stages, these routines will also eventually be modified.

4. Performance Analysis

The small diode described in Figures 1 and 2 was simulated using Parallel PISCES. Because of size restrictions placed upon the PISCES program by the host processor's XENIX operating system (see section 2), this was the largest device that could be simulated with the current implementation. Figure 4 plots the total run time of the simulation as an accumulation of the time spent transferring the state of the simulation to the hypercube and of the time spent executing the LU factorization, triangular solution, and matrix assembly routines in the Newton loops. These are plotted against the dimension of the hypercube on which the simulation was run. A comparison of the data with Table 1 shows that a one processor hypercube runs four percent slower than the host. Most of this difference can be attributed to the transfer of data between the host and the hypercube. The remainder can be attributed to the different ordering strategies used on the sparse matrices. The host processor uses a minimum degree heuristic while the hypercube uses a nested dissection. On a rectangular grid, such as the one in Figure 1, the minimum degree routine effectively performs a cyclic reduction of the grid thus creating a better ordering than the nested dissection. On non-rectangular grids, the cyclic reduction is not as effective and the nested dissection is expected to produce a better ordering.

The time required to perform the symbolic decomposition by the host processor was almost identical to that required by a one processor hypercube.

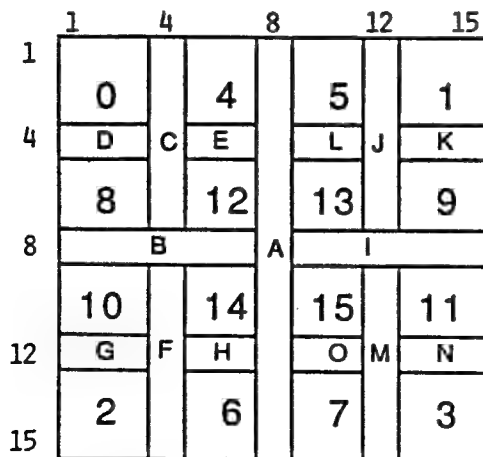


Figure 3
Distribution of 15 by 15 grid to 16 processors
The number in each block corresponds
to the address of its processor

This is surprising in that the host uses a minimum degree ordering heuristic that is more complicated than the nested dissection used by the hypercube. However, the hypercube performs a more complicated symbolic decomposition in that it has to identify separators and treat them as dense subproblems. These differences appear to balance well in the diode example presented. For larger problems, the hypercube is expected to run faster. When the dimension of the hypercube increases, a significant speedup is observed. One processor required 30.2 seconds to perform the ordering and symbolic decomposition. Sixteen processors took only 4.5 seconds.

As the number of processors in Figure 4 is increased, the time required to assemble the problem initially drops dramatically. Four processors assemble the problem in 27% of the time required by only one. This is because the vertices of the grid are evenly divided amongst the processors and the load is almost perfectly balanced for the assembly operation. As the number of processors increases to 8 and then 16, the vertices are still evenly divided, but the triangles are not. The processors in the center of the grid have as many as 32 triangles where as those on the corners can have as few as 18. This skews the load balance during the assembly phase and limits the speedup to a factor of 6.7 as compared to the host. On larger problems, an even distribution of the vertices should result in a more even distribution of the triangles. Therefore speedups approaching the number of processors should be possible.

Again, as the number of processors in Figure 4 is increased, the time required to factor the sparse matrices and perform the forward and back substitutions is decreased. Speedup of the factorization phase, relative to the host, is 1.7 for two processors but only 4.9 for 16 processors. This is not as disappointing as it first appears. In fact, it is quite predictable. The problem being simulated is based on a 15 by 15 grid and contains three variables at each vertex. Therefore, the entire system contains only 675 equations. Figure 5 is a plot of the speedup achieved by the DMF sparse matrix factorization algorithm using 4, 8, and 16 processors to factor matrices derived from the application of a five point stencil to square grids. Figure 6 shows the speedups of the corresponding triangular solvers. A vertical line has been introduced to highlight the size of the matrices used in the diode example. It is clear from Figure 5 that a speedup of less than five was to be expected for the sparse matrix factorization. A measure of just how small this problem really is can be seen by the performance of the triangular solvers. Their speedup peaks for four processors and they actually run slower for 8 and 16. This is because there is not enough floating point arithmetic to overcome the increased communication required as more processors are added to the solution. The triangular solvers are thus communication bound.

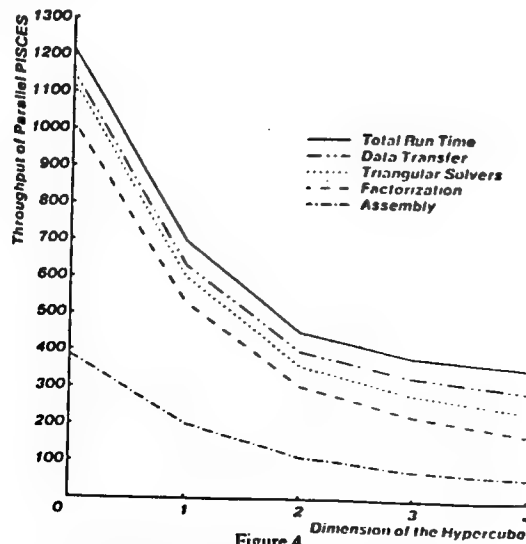


Figure 4
Run time of simulation defined in Figure 2
Total shown as an accumulation of its major parts

There is one last observation to be made about Figure 4. The time required to transfer data to and from the hypercube increases from 2% to 15% of the total time required to run the simulation. This is a manifestation of the observation that a supercomputer merely transforms compute bound problems to I/O bound ones. As the size of the problem increases, the data transfer percentage will decrease. However, it will increase with the number of processors. Ultimately, the only way to limit the data transfer problem will be to implement a larger portion of the program on the parallel processor.

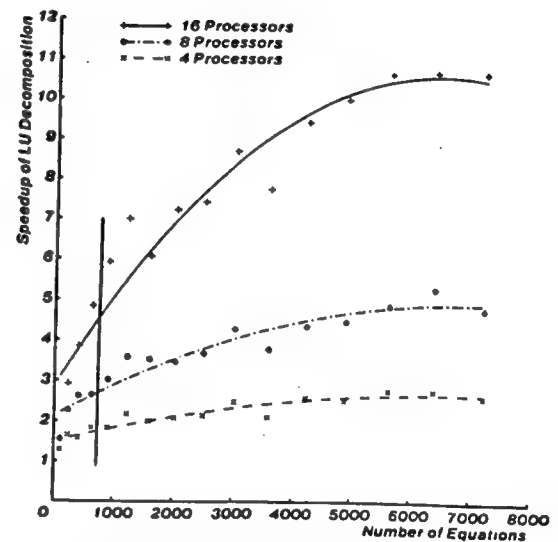


Figure 5
Speedup of distributed multifrontal sparse matrix factorization

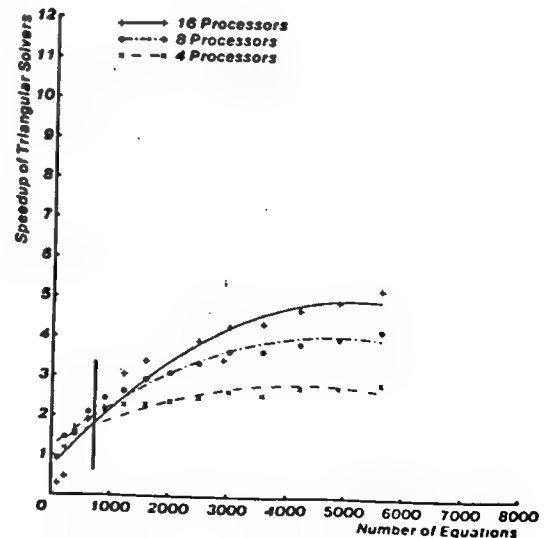


Figure 6
Speedup of distributed multifrontal forward and back substitution

5. Conclusion

A parallel implementation of Stanford's PISCES two dimensional device simulator has been presented. It demonstrates an entire solution of the system of non-linear equations that model a device's behavior, including matrix ordering and symbolic decomposition, on a distributed memory hypercube. The performance of the system was consistent with the performance of the DMF sparse matrix factorization routines depicted in Figures 5 and 6. The example presented was restricted in size due to limitations in the host processor. It clearly showed the price of inadequate load balancing and increasing communication brought on by the application of 8 or 16 processors to a small problem. Future implementations of Parallel PISCES will overcome these limits and solve much larger problems. It is clear from Figure 5 that an implementation of PISCES that solves systems of 4000 or more equations, the size of a typical problem, can expect to achieve speedups of ten using 16 processors. This corresponds to a parallel efficiency of 62%. An example of such a problem is the CMOS trench isolation example in the PISCES technical report. Figure 7 is taken from this example. It contains 1303 vertices in its grid and requires the solution of systems of 3909 equations.

Achieving peak efficiency requires static load balancing be performed during the discretization of the grid. The optimum balance for DMF matrix factorization requires that the processors on the corners of the grid contain more vertices than those in the center². This conflicts with the equal distribution required for peak speedup of the matrix assembly operation. Fortunately, this problem is limited to hypercubes with small numbers of processors. As the number of processors reaches 64 (a six dimensional hypercube), most of the processors are assigned blocks in the interior of the grid, and an even distribution of the grid to the processors would not seriously degrade the performance of the DMF factorization.

Parallel PISCES has been presented as an answer to the computational bottleneck facing device designers. The parallel implementation of the ordering and symbolic decomposition of the simulation grid has yielded significant speedups as well as decreased the data traffic necessary between the host and the multiprocessor. Parallelizing the assembly and solution of the sparse matrices in the Newton loops of the SOLVE card has been shown to yield significant speedups. As the size of the problems grow and more of the overall program is implemented concurrently, parallel efficiencies of 60% to 70% should be easily achieved.

Acknowledgments

The primary source of financial support for this project has been the U.S. Army Research Office, Contract No. DAAG29-83-K-0125. Additional financial and material support was provided by the General Electric Company and by Intel Corporation. This work was motivated by the insight of Dr. Jerome Tiemann and greatly facilitated by the patient assistance of Conor Rafferty.

References

- [1] Mark R. Pinto, Conor S. Rafferty, and Robert W. Dutton : "PISCES-II: Poisson and Continuity Equation Solver", Stanford Electronics Laboratory, Tech. Report, Sept. 1984
- [2] Robert Lucas, Tom Blank, and Jerome Tiemann "A Parallel Solution Method for Large Sparse Systems of Equations" *IEEE International Conference on Computer-Aided Design* 1986, pp. 178-181
- [3] Conor S. Rafferty, Mark R. Pinto, and Robert W. Dutton : "Iterative Methods in Semiconductor Device Analysis", *IEEE Transactions on Computer Aided Design*, Vol. CAD4, No. 4, Oct. 1985, pp.462-471
- [4] private conversation, Conor S. Rafferty and Mark R. Pinto : EE Dept., Stanford University, June 1984
- [5] H. T. Kung and C. E. Leiserson : *Algorithms for VLSI Processor Arrays*, Addison-Wesley, Reading, MA, 1980, pp. 271-292
- [6] James M. Ortega and Robert G. Voigt : "Solution of Partial Differential Equations on Vector and Parallel Computers", *SIAM Review*, Vol. 27, No. 2, June 1985, pp. 149-240
- [7] O. Wing and J. W. Huang : "A Computation Model of Parallel Solution of Linear Equations", *IEEE Transactions on Computers*, Vol. C-29, July 1980, pp. 632-638
- [8] Iain S. Duff and J. K. Reid : "The Multifrontal Solution of Indefinite Sparse Symmetric Linear Equations", *ACM Transactions on Mathematical Software*, Vol. 9, No. 3, Sept. 1983, pp. 302-325
- [9] Gita Alghband and Harry F. Jordan : "Multiprocessor Sparse L/U Decomposition with Controlled Fill-in", Tech. Report 85-48, ICASE, NASA Langley Research Center, Hampton, VA 23665 (1985)
- [10] George Jacob, et. al. : "Direct-Method Circuit Simulation Using Multiprocessors", *IEEE International Symposium on Circuits and Systems*, 1986, pp. 170-174
- [11] George A. Geist and Michael T. Heath : "Parallel Cholesky Factorization on a Hypercube Multiprocessor", Tech. Report ORNL/TM-9962, Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831 (1985)
- [12] J. A. Clinard and G. A. Geist : "Implementing Fracture Mechanics Analysis on a Distributed Memory Parallel Processor", Tech Report ORNL/TM-10367, Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831 (1987)
- [13] Craig H. Price : Ph.D Thesis, Stanford University, May 1982
- [14] R. S. Varga : *Matrix Iterative Analysis*, Prentice-Hall, Englewood Cliffs, NJ, 1962
- [15] Gene Golub and Charles Van Loan : *Matrix Computations*, The Johns Hopkins University Press, Baltimore, MD, 1983
- [16] D. J. Rose : "A Graph-Theoretic Study of the Numerical Solution of Sparse Positive Definite Systems of Linear Equations", *Graph Theory and Computing*, edited by R. C. Read, Academic Press, New York, NY, 1972
- [17] Cleve Moler and David Scott : "Communication Utilities for the iPSC" iPSC Tech. Report #1, Intel Scientific Computers, Beaverton, OR
- [18] Alan George and Joseph Liu : *Computer Solution of Large Sparse Positive Definite Systems*, Prentice Hall, Englewood Cliffs, NJ, 1981
- [19] Cleve Moler : Second Conference on Hypercube Multiprocessors, Knoxville, TN, Sept. 1986

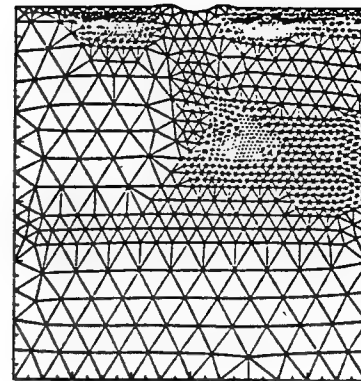


Figure 7
CMOS Trench Isolation simulation grid
Taken from PISCES Technical Report

A Parallel Solution Method for Large Sparse Systems of Equations

R.F. Lucas

T. Blank

J.J. Tiemann

Reprinted from
IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN
Vol. CAD-6, No. 6, November 1987

A Parallel Solution Method for Large Sparse Systems of Equations

ROBERT F. LUCAS, TOM BLANK, AND JEROME J. TIEMANN, FELLOW, IEEE

Abstract—This paper presents a new distributed multifrontal sparse matrix decomposition algorithm suitable for message passing parallel processors. The algorithm uses a nested dissection ordering and a multifrontal distribution of the matrix to minimize interprocessor data dependencies and overcome the communication bottleneck previously reported for sparse matrix decomposition [1]. Distributed multifrontal forward elimination and back substitution algorithms are also provided. Results of an implementation on the Intel iPSC are presented. Up to 16 processors are used to solve systems with as many as 7225 equations. With 16 processors, speedups of 10.2 are observed and the decomposition is shown to achieve 67 percent processor utilization. This work was motivated by the need to reduce the computational bottleneck in the Stanford PISCES [2] device simulator; however, it should be applicable to a wide range of scientific and engineering problems.

I. INTRODUCTION

A SYSTEM OF N equations in N unknowns can be represented as a matrix equation $Ax = b$, where the vector x contains the unknowns, the matrix A contains their coefficients, and the vector b contains the right-hand sides of the equations. Assuming it is of full rank and that a direct rather than iterative solution is desired, the matrix A could be inverted and x computed by the product of $A^{-1} * b$. Unfortunately, matrix inversion is computationally expensive. Furthermore, even if the matrix A is sparse, A^{-1} is generally dense, thus limiting the size of the problems that can be solved. Because of these problems, the matrix A is usually factored by LU decomposition into two triangular matrices. The resulting triangular system is then easily solved.

Sparse LU decomposition plays an extremely important role in the simulation of physical phenomenon. For example, it can account for 90 percent of the run time of a numerical device simulation using the Stanford PISCES program [2], [3]. Furthermore, as the size of the matrix being factored increases, the turnaround time of sparse LU decomposition grows super-linearly. The growth in problem size is currently out-pacing improvements in the conventional von Neumann-style computers traditionally used to solve such problems. Fortunately, LU decompo-

sition contains inherent concurrency that can be exploited to improve its throughput in parallel processing environments. Many efforts are being made to utilize this concurrency on existing parallel machines [1], [4]–[7].

Large ensembles of inexpensive VLSI processors have been proposed that could exploit the concurrency available in sparse matrix decomposition [8], [9]. The systolic array offers an extremely cost effective approach to decomposing banded matrices. However, it is limited to solving problems whose bandwidth is a function of the physical size of the array itself. Furthermore, it fails to address related issues such as matrix ordering and assembly. To address all of the aspects of problems such as device simulation, an ensemble of general-purpose processors is required.

Available parallel processors can be classified as being either multiple instruction, multiple data stream (MIMD) or single instruction, multiple data stream (SIMD). SIMD machines offer tremendous speedups when applied to nested Fortran DO loops that do not contain insurmountable data dependencies. Unfortunately, potentially concurrent portions of applications programs are often not incorporated in nested loops and cannot be parallelized. The scalar control processor becomes the bottleneck, and disappointing speedups are attained [10].

MIMD machines are more flexible in that different processors can execute unrelated code segments concurrently. The problem with MIMD systems is interprocessor synchronization and communication. For a system with a small number of processors, such as the four processor Cray X-MP4/8¹, it is feasible to have the processors communicate via shared registers. For larger shared memory systems, synchronization is accomplished through shared variables or semaphores resident in the global memory. On the X-MP4/8, the functional equivalent of a P or V semaphore operation can take hundreds of clock cycles [11], [12]. This is a tremendous delay when compared to the throughput of the machine's vector arithmetic processors which can produce a double precision result every clock cycle. The communication cost is even higher for message passing systems where the delays of formatting and propagating the message through a network must also be included. For example, on the Intel iPSC², transmitting a message between adjacent proces-

Manuscript received October 20, 1986; revised May 28, 1987. This work was supported by the U.S. Army Research Office, Contract No. DAAG29-83-K-0125.

R. F. Lucas is with the Integrated Circuits Laboratory, Stanford University, Stanford, CA 94305.

W. T. Blank is with the Center for Integrated Systems, Stanford University, Stanford, CA 94305.

J. J. Tiemann is with the General Electric Corporate Research and Development Center, Schenectady, NY 12301. He is also a Consulting Professor at Stanford University.

¹X-MP is a trademark of Cray Research Incorporated.

²iPSC is a trademark of Intel Corporation.

sors takes 1 ms. Thus, it is crucial for applications programs that seek to use large-scale parallel processors to minimize interprocessor communication.

This paper presents a new distributed multifrontal (DMF) sparse matrix decomposition algorithm³ for the parallel processing environment. It uses a nested dissection ordering and multifrontal distribution of the matrix to minimize interprocessor data dependencies and overcomes the communication bottleneck reported for general sparse solvers [6]. The paper is organized as follows: Section II reviews sparse LU decomposition and highlights the concurrency that can be exploited. Section III introduces the DMF method of sparse matrix decomposition. Section IV describes the triangular solvers used in conjunction with the DMF decomposition. In Section V, the performance of the DMF algorithm is analyzed, and results from an implementation on the Intel iPSC/MX are presented. Finally, conclusions are drawn and applications of this work are suggested.

II. LU DECOMPOSITION

A matrix is reduced to an upper triangular form U by performing a series of Gauss eliminations [14]. These are linear operations whereby each pivot row k is subtracted from each succeeding row j rendering all of the elements a_{jk} zero. The resulting triangular system is easily solved by back substitution. If the ratios of the pivot rows that were used to perform the eliminations are preserved in the locations of the lower triangular elements that were eliminated, multiple systems with the same coefficient matrix can be solved. The resulting lower triangular matrix is L and the processes of factoring the matrix is called LU decomposition.

An algorithmic description of LU decomposition is provided in Fig. 1. The outer-most loop selects the pivot elements from the diagonal. Within the outer loop, the two basic operations are Divide (statement labeled 1 in Fig. 1) and Update (statement labeled 2 in Fig. 1). Each of these steps contains potential concurrency that can be exploited to improve the throughput of the matrix factorization. If there are m non-zero elements of column L_k , there are m independent floating-point operations (FLOP's) in the Divide step. If the matrix is symmetric, there are m^2 floating-point multiply-accumulate operations (two FLOP's per operation) in the Update step. If no attempt is made to exploit sparsity, then the algorithm is easily parallelized by assigning columns of the matrix to processors in a wraparound fashion [15]. The Divide operation is vectorized and executed by the processor that contains the pivot element. The resulting column of L is then broadcast to all of the other processors. Each processor computes the updates to its columns of the matrix independently. Fig. 2 provides an algorithmic description for a message passing multiprocessor.

³This paper assumes the reader has some prior knowledge of sparse matrix problems. An excellent reference is *Computer Solution of Large Sparse Positive Definite Systems* [13] by George and Liu.

Let : $A = [a_{i,j}]$
 $L = [l_{i,j}]$
 $U = [u_{i,j}]$ } $1 \leq i, j \leq n$

such that L is lower triangular, U is upper triangular and

$$A = LU$$

Algorithm GE is :

```

do 4 k=1,n
do 1 j=k+1,n           {Create column Lk}
1  lj,k = -aj,k/ak,k
do 2 i=k+1,n           {Update Ak submatrix}
do 2 j=k+1,n
2  ai,j = ai,j + ak,j * li,k
do 3 j=1,k-1           {Store column Uk}
3  uj,k = aj,k
4 continue

```

Fig. 1. Column oriented Gaussian elimination.

Let : n be the rank of the submatrix
 NP be the number of processors
 ME be the identifier of the local processor ($0 \leq ME < NP$)

Algorithm Concurrent GE is :

```

do 4 k=1,n
if ((k modulus NP) = ME) then
{The pivot resides on this processor}
do 1 j=k+1,n           {Create column Lk}
1  lj,k = -aj,k/ak,k
Broadcast Lk
else
{The pivot resides on another processor}
Await receipt of Lk
end if
do 2 i=k+1,n           {Update Ak submatrix}
do 2 j=k+1,n
2  ai,j = ai,j + ak,j * li,k
do 3 j=1,k-1           {Store column Uk}
3  uj,k = aj,k
4 continue

```

Fig. 2. Concurrent message passing Gaussian elimination.

For solving sparse matrices, a lower bound on the number of operations required was derived by Wing and Huang [16]. An algorithm has since been proposed that would schedule the directed acyclic graph that describes this matrix solution on a multiprocessor system [17]. However, this algorithm assumes an unbounded parallel model where multiple processors may simultaneously access the same location in a global memory without contention. Unfortunately, there are no real machines available that implement this abstract model of computation.

A more practical algorithm for solving general sparse systems has been implemented [6] in which the elimination tree of the sparse matrix is used to identify pivots that can be processed independently. An example of the elimination tree resulting from a nested dissection ordering of a 3 by 3 grid is provided in Fig. 3. The elimination tree is a representation of the data dependencies between the pivots of the matrix. Nodes of the tree cannot be eliminated until after the leaves below them. Columns of the sparse matrix are assigned to processors in a wraparound fashion by traversing the elimination tree from its leaves to its root. The processors that contain leaves of the tree are free to perform the Divide operation on those columns. The resulting columns of L are then transmitted to the processors that need them to update their columns. The eliminated columns are removed from the tree and new leaves are exposed.

An alternate distribution of the sparse matrix has been

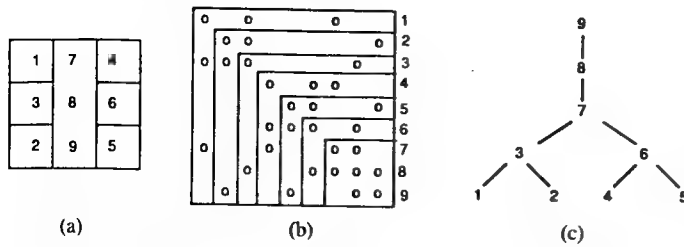


Fig. 3. Nested dissection ordering, matrix non-zero structure, and elimination tree for 3×3 grid.

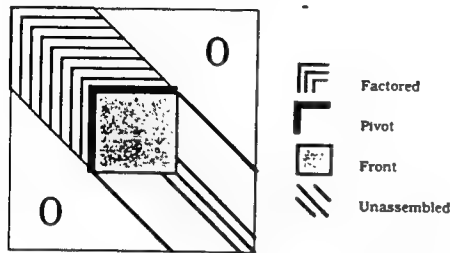


Fig. 4. Frontal decomposition of band matrix.

proposed by Duff and Reid [5]. It is motivated by the frontal or out-of-core matrix factorization technique. The frontal method [5], [18] was introduced as a means of solving finite-element systems that are too large to reside in the main memory of a computer. Fig. 4 contains a representation of the factorization of a band matrix by the frontal method. The submatrix representing a physically adjacent set of elements is assembled and factored. Updates generated for locations in the matrix beyond those already assembled are maintained in a separate submatrix called the front. After elements are decomposed, the factors are placed in secondary storage. New elements are then assembled with updates from the values stored in the front. The procedure continues until the entire matrix has been factored.

In their multifrontal work, Duff and Reid have observed that the frontal method can be applied concurrently to the leaves of the elimination tree. Each processor assembles the submatrix corresponding to the row and column of a leaf of the elimination tree along with the resulting front. The processor can independently perform the Divide operation on its column and update its front. Multiple processor's fronts can overlap, effectively decoupling the multiplication and addition operations in the Update steps. While this prevents chaining of the arithmetic units of the processors, it still provides for long vectorized multiplications. Furthermore, partial updates of the same location can be accumulated concurrently. An implementation has been proposed for the Denelcor HEP. Processors synchronize and resolve data dependencies by communicating through the shared memory. For distributed memory systems, Duff has suggested that an increase in the granularity of the problem is warranted. This could be accomplished by assigning branches of the elimination tree to individual processors.

The emphasis in the previous work has been to identify arithmetic operations that can be processed concurrently.

Interprocessor communication has been either ignored [16], [17] or accepted as an overhead of the algorithm. The only attempt to reduce the communication has focused on the ordering of the matrix and the resulting distribution of columns to processors [19]. This technique uses the previously described general sparse algorithm and the reduction in messages will be limited to a factor of $\log_2 N$ provided by the ordering.

III. DISTRIBUTED MULTIFRONTAL LU DECOMPOSITION

This work differs from previous work in that an attempt is made to maximize computational throughput by minimizing the communication overhead. A distributed multifrontal sparse matrix decomposition algorithm is presented in which communication is restricted by deferring the resolution of interprocessor data dependencies. Each processor accumulates multiple updates to locations that reside on other processors within a local front. An increase in the volume of storage needed to solve the problem is traded for a decrease in the number of messages exchanged. Before proceeding with the discussion of the DMF matrix factorization, a few definitions are necessary.

Block will always refer to a block of the dissected problem (see Fig. 5).

A *pivot element* is an element, $a_{k,k}$, on the diagonal used to eliminate all lower triangular elements in column k , $a_{j,k} | j > k$.

The *pivot $a_{k,k}$'s row* will be the elements of the upper triangle in row k , $a_{k,j} | j > k$.

The *pivot $a_{k,k}$'s column* will be the elements of the lower triangle in column k , $a_{j,k} | j > k$.

A *block's rows and columns* will be the rows and columns of the pivots within the block.

Similarly, a *separator's rows and columns* will be the rows and columns of the pivots within the separator.

Elements of x_k , y_k , and b_k will be considered *associated* with pivot $a_{k,k}$ and the block or separator that contains the pivot.

A block or separator's *submatrix* is the set of all locations in the matrix contained in the rows and columns of the block or separator and the front consisting of the locations updated while factoring them.

An *off-diagonal column* of U , U_k , of a block or separator is any column of the submatrix such that $a_{k,k}$ is not a pivot contained in that block or separator.

The best implementation of any algorithm is a function of the target architecture. Throughout the remainder of this paper, the parallel processor shall be assumed to be a distributed memory message passing MIMD hypercube. There are currently four such machines commercially available. Each processor contains a communication channel to every other processor whose binary identifier differs by only one digit. Therefore, the hypercube architecture requires only $\log_2 P$ communication ports per processor, where P is the total number of processors. These machines lend themselves to massively parallel implementations.

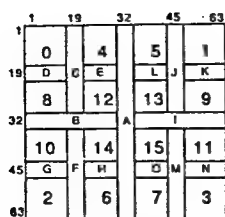


Fig. 5. Incomplete nested dissection of 63×63 grid. The number of each block corresponds to the address of its processor. Letters label the separators.

The DMF technique was designed to reduce the communication overhead. The number of potential messages is minimized by assigning pivots to processors on the basis of spatial locality in the dissected problem. This is accomplished by using a nested dissection [13] of the underlying problem grid. At each stage of the dissection process, a separator is found that divides the problem grid into two blocks. This procedure is recursively applied to each block until a block has been isolated for every processor.

Conceptually, the entire grid initially resides in processor zero. Processor zero begins the dissection process by identifying a separator and dividing the grid into two disjoint blocks. The elements of the separator are ordered so as to be factored last. Processor zero then transmits one of the blocks to an adjacent processor. The procedure is repeated $\log_2 P$ times, where P is the number of processors. At each step i , $0 \leq i \leq \log_2 P - 1$, a processor x divides its block between itself and processor $x + 2^i$. An example of the resulting distribution of a 63 by 63 grid over 16 processors is given in Fig. 5, where letters are used to label the separators and numbers identify the processors in which each block resides.

Within its block, each processor orders the matrix independently. A processor with scalar arithmetic units can continue the nested dissection to produce an ordering that minimizes storage and operation counts. A vector processor can terminate the dissection process and instead use a minimum bandwidth ordering strategy within its block. This process, called incomplete nested dissection [20], creates a matrix suitable for vector processing. An example of an incomplete nested dissection of a 7 by 7 grid is given in Fig. 6. The structure of the resulting matrix is shown in Fig. 7. The subregions of Fig. 7 correspond to the rows and columns of the blocks and separators of Fig. 6.

Each processor locally assembles the submatrix consisting of the rows and columns of its block and the front corresponding to the separators that were used to isolate the block. Fig. 8 shows the subsets of Fig. 7 that would be stored in each of the four processors. It also illustrates the redundant storage of the fronts associated with the separators. The blocks can be factored without interprocessor communication. Interprocessor data dependencies manifest themselves in updates to the front. Updates to the elements in the separator fronts are stored in the processors that generate them.

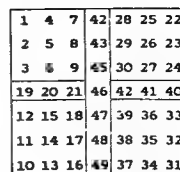


Fig. 6. Incomplete nested dissection ordering of a 7×7 rectangular grid.

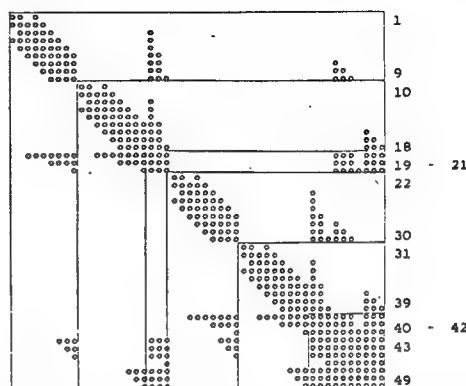


Fig. 7. Incomplete nested dissection matrix structure for 7×7 grid.

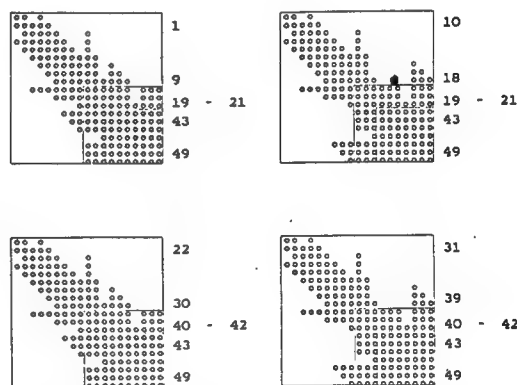


Fig. 8. Four processor distributed multifrontal distribution of the Fig. 7 matrix.

After the blocks have been factored, processors whose blocks were isolated by one of the separators cooperatively factor that separator's submatrix. They do so using the message-passing algorithm in Fig. 2 modified so that the outer loop terminates before selecting any pivots from the front. In Fig. 5, separator D would be factored by processors 0 and 8. The separator's submatrix is composed of updates generated by both processors as well as the initial values of the matrix. Before factoring of D can commence, these values must be accumulated. As each processor is allocated every other column of the dense separator submatrix, updates to a column resident on another processor must be transmitted to that processor. This results in the exchange of every column of the submatrix between two adjacent processors in the hypercube. These are long efficient messages in that the overheads associated with formatting and transmitting the message are amortized over many floating-point numbers. For the Intel iPSC, messages are transmitted in packets of 1024 bytes.

Thus, 128 double precision numbers can be transmitted at the cost of formatting one packet. After the columns are exchanged, the rows and columns of the separator are factored.

This process is recursively applied to all of the separators. Each time, the number of cooperating processors doubles. Separator *C* of Fig. 5 is factored by processors 0, 4, 8, and 12. Again, partial updates of each column of the separator's submatrix generated while factoring separators *D* and *E* must be accumulated. To minimize data traffic, columns of separator *C*'s submatrix are assigned to processors that contained the same columns while factoring the preceding separators. If the columns are allocated to the processors in the order 0, 8, 4, 12, then processors 0 and 8 exchange columns while processors 4 and 12 do. The number of messages that have to be exchanged is limited to one per column of the submatrix and communication is restricted to nearest neighbors in the hypercube.

To clarify the DMF algorithm, a detailed example is provided in Fig. 9. The figure has 14 parts representing the various steps four processors would take to solve a system of 25 equations. Fig. 9(a) illustrates the dissection, ordering, and allocation of the elements of a 5 by 5 grid to the four processors. Fig. 9(b) contains a sample matrix built upon the adjacency structure defined in Fig. 9(a). To illustrate the sparsity of the matrix, only nonzero elements are displayed. Fig. 9(c) details the initial distribution of the blocks from Fig. 9(b) over four processors. While processors 0 and 1 contain different parts of the matrix, their local blocks have an identical structure that appears only once. The same is true for processors 2 and 3. Fig. 9(d) shows the factored blocks and the updates made to the separator fronts. Zeros that result from arithmetic operations are displayed to illustrate the updates. In contrast, the initial nonzeros in the fronts are not included in Fig. 9(c) and (d). This serves to illustrate both the fill and the fact that the initial values need not be added to the fronts until the fronts themselves are factored. Fig. 9(e) contains the initial separator *B* and *C* submatrices and shows the allocation of the columns to the processors. Fig. 9(f) contains the factored submatrices and the updates to their fronts. Fig. 9(g) shows the initial separator *A* submatrix and the allocation of its columns to all four processors. Fig. 9(h) shows the factored submatrix. Fig. 9(i)–(n) detail the solution of the resulting triangular systems and will be discussed in the next section.

IV. DISTRIBUTED MULTIFRONTAL FORWARD AND BACK SOLVERS

Matrix decomposition is only part of the solution of a system of equations. The triangular systems $Ly = b$ and $Ux = y$ must also be solved. The first is solved by forward elimination (FE), while the latter is solved by back substitution (BK). While decomposition may be the more computationally intensive, a parallel implementation of the triangular solvers is also required.

FE is the process of updating the vector *b* by the same

linear transformations used to factor the matrix *A*. Each column *j* of *L* is multiplied by each element b_j and subtracted from the vector *b*. Fig. 10 contains an algorithmic description of forward elimination. The outer loop selects the column that will be used to update *b*. The inner loop contains the multiply-accumulate operation and is easily vectorized. However, there is no equivalent to the Update step of LU decomposition with its multiple vector operations. In fact, FE is merely an extension of the Update step of the LU decomposition applied to *b*. Thus, there is less concurrency to exploit.

To perform FE, each processor assembles the elements of *b* that correspond to its block of the dissected problem. As FE is an extension of the Update step of the matrix factorization, columns of *L* within the blocks of the dissected problem can update *b* without interprocessor communication. Again, updates to locations in *b* that correspond to separators of the problem grid are stored in the processors that generated them.

Factorization of the separator submatrices leaves the columns of the separators in *L* distributed over multiple processors. Therefore, the processors that factored a separator must cooperate to perform FE with its columns. As the multifrontal distribution of the matrix leaves entire columns of *L* on one processor, the processor that contains the first column of the separator, L_j , must receive the other processor's updates to *b*. It adds them to its own updates and then transforms *b* with column L_j . The subset of *b* associated with the separator and its front is passed between the processors until all columns of *L* in the separator have transformed *b*. If this is not the last separator, the processor containing the final column in the separator transmits the updates to *b* in the separator's front to the processor that contains the first column of the next separator.

The detailed example in Fig. 9 includes forward elimination. Fig. 9(i) shows the blocks of *L* and the initial distribution of *b*. It also contains the resulting subsets of *y* after FE by the blocks of *L*. Fig. 9(j) and (k) detail the transformation of *b* by the separators of *L*. Fig. 9(j) contains elements of *b* after assembly and accumulation of the updates generated by the blocks. It also contains the resulting elements of *y* after FE is performed with the columns of separators *B* and *C*. Fig. 9(k) continues the example for separator *A*.

The back substitution phase solves the upper triangular system $Ux = y$ to generate the solution vector *x*. The last element of *y* (i.e., y_n) is divided by $u_{n,n}$ computing x_n . The entire column U_n can now be multiplied by x_n and subtracted from *y*. This reduces the problem to an upper triangular system of $n - 1$ equations. The process is repeated until a solution is found for every element of *x*. An algorithmic description is provided in Fig. 11.

The DMF implementation of BK is complicated by the fact that entire columns of *U* do not reside in one processor. This stems from the fact that multiple fronts overlap. While two processor's blocks may update the same column of a front, each column of the submatrix resides on


```

Let:  $L = \{l_{ij}\}$ 
       $y = \{y_i\}$ 
       $b = \{b_i\}$ 
}  $1 \leq i, j \leq n$ 

such that  $L$  is lower triangular and

 $Uy = b$ 

Algorithm FE is:

do 2  $i = 1, n$ 
   $y_i = b_i$ 
  do 1  $j = i + 1, n$ 
    1  $b_j = b_j - y_i * l_{ji}$ 
  2 continue

```

Fig. 10. Column oriented forward elimination.

```

Let:  $U = \{u_{ij}\}$ 
       $y = \{y_i\}$ 
       $x = \{x_i\}$ 
}  $1 \leq i, j \leq n$ 

such that  $U$  is upper triangular and

 $Ux = y$ 

Algorithm BK is:

do 2  $i = n, 1, -1$ 
   $x_i = y_i / u_{ii}$ 
  do 1  $j = 1, i - 1$ 
    1  $y_j = y_j - x_i * u_{ji}$ 
  2 continue

```

Fig. 11. Column oriented back substitution.

only one of the processors when the front is factored. An example is column 21 of Fig. 9(b). Nonzero elements of column 21 reside in blocks of the dissected problem as well as in separator submatrices. The column of U is distributed over processors 0 and 1.

BK is begun by allowing the processor that contains the final column of U in the last separator submatrix to solve for x_n . It then multiplies x_n by the elements of U_n that reside in the separator submatrix and subtracts them from y . The separator's subset of y is then transmitted to the next processor which solves for x_{n-1} . BK is restricted to the dense separator submatrix for two reasons. First, the processors that performed BK on the dense separator submatrix do not contain entire columns of U . Also, as only one processor is active at a time, it is imperative that this submatrix is exited as quickly as possible and subsequent submatrices, where greater concurrency is possible, be entered.

When the elements of x corresponding to a separator have been solved, they are broadcast to all of the processors that factored the separator. These processors are the ones that factored the preceding two separators and they contain the columns of U allocated to the two separators. The values of x that have already been computed are multiplied by the off-diagonal columns of U and subtracted from the separator's elements of y . The elements of x corresponding to the separators are then computed as above. The procedure is applied recursively to all preceding separators.

When the BK has been performed with all of the separators, the processors each contain the solutions of the elements of x corresponding to the separators that isolated their block. They are free to compute the solutions of the elements of x in their blocks without communication. The final solution vector x is distributed over the processors that computed it.

Fig. 9 concludes with an example of back substitution. Fig. 9(1) shows x and y both before and after the solution of x_{21} through x_{25} . Fig. 9(m) details the transformation of y and the computation of values of x by separators B and C . Fig. 9(n) shows the blocks of U and initial elements of x and y needed to independently compute the remaining components of x . It also contains the final values of x stored with each block.

Like matrix factorization, FE and BK both enable large portions of the algorithm to be processed independently. However, when manipulating the separators, only one of the cooperating processors can perform arithmetic operators at any time. In fact, when performing FE or BK on the last separator in the matrix, only one of the processors in the multiprocessor is active at a time. Therefore, smaller speedups are to be expected than those achieved during the factorization.

V. PERFORMANCE ANALYSIS

The DMF algorithm allows the processors to factor all of the rows and columns of the matrix, except those of the separators, without interprocessor communication. For a rectangular block ordered using incomplete nested dissection, the number of floating-point operations required is αm^2 , where m is the half-bandwidth of the block and α is a function of the number of separator nodes adjacent to the block. Detailed equations are derived in the Appendix. For a separator S , factorization requires $(2/3)x^3 - 2bx^2 + 3b^2x$ FLOP's, where x is the length of the separator and b is the number of nodes of other separators that bound the block S divides. In the above equations, m , x , and b are $O(N^{0.5})$, where N is the number of equations in the matrix. Therefore, the work performed in the blocks scales as $O(N^{2.0})$, whereas the work for the separators scales as $O(N^{1.5})$. For large problems, where the number of equations is much greater than the number of processors, most of the work can be performed without interprocessor communication. Tables I and II provide examples. Table I details the workload distribution for factorization of the blocks of a 63 by 63 grid dissected as shown in Fig. 5. The length of the shorter side of the block is m_s , while the length of the longer side is m_l . Note from Fig. 5 that the blocks are not of equal size. The number of nodes in each block of the grid has been adjusted to reflect the fact that the work performed in each block is a strong function of the number of adjacent separator nodes (see the Appendix). This static load balancing is performed by the nested dissection ordering heuristic. The 40-percent difference in the work reported for the processors in Table I is a tremendous improvement over the factor of four that would exist if the blocks were all of equal size. Table II details the work required to factor the separators of the same problem. Again, notice that the lengths of the separators, and thus the work required to factor them, varies. The total work shown at the bottom of Tables I and II shows that 66 percent of the work is involved in factoring the blocks and can be performed before any messages are exchanged. Therefore, even if there is a substantial overhead for interprocessor communication, significant speedup is readily achievable.

TABLE I
DISTRIBUTION OF WORK AMONG BLOCKS OF FIG. 5

Block	m_1	m_2	FLOPs (10^3)/Block
0,1,2,3	18	18	379
4,5,6,7,8,9,10,11	12	18	446
12,13,14,15	12	12	531
Total FLOPs			7208

TABLE II
DISTRIBUTION OF WORK AMONG THE SEPARATORS OF FIG. 5

Separator	x	b	FLOPs (10^3)/Separator
D,L,G,O	17	48	115
E,H,K,N	13	75	183
C,F,J,M	31	62	386
B,I	31	63	393
A	63	0	167
Total FLOPs			3689

In Tables III and IV, the DMF method of sparse matrix decomposition is compared to the general sparse (GS) method reported by Giest of assigning columns of the sparse matrix to processors based upon the elimination tree. The two parallel algorithms are also compared to the serial LU decomposition algorithm used by PISCES. Measurements were taken on an Intel iPSC/MX hypercube multiprocessor. Each processing node can sustain a peak arithmetic throughput of 32 KFLOP's. The minimum time required for transmitting a message between two adjacent processors is between 1 and 3 ms, depending upon the operating system version. Table III contrasts the communication loads of the DMF method and a general sparse Cholesky solver provided by E. Ng of the Oak Ridge National Laboratory. The GS algorithm broadcasts the pivot column to each processor that contains dependent columns. Therefore, the number of messages is a function of the density of the sparse matrix. A nine-point stencil requires more messages than a five-point stencil. In contrast, the number of messages sent by the DMF algorithm is a function of the length of the separators. The message count is independent of both matrix density and the ordering of the local blocks. As a result, the GS method has a much greater communication load than does the DMF method. The price of a higher communication load of the GS method is demonstrated in Table IV. The DMF algorithm solves the five-point 961-line system of equations 2.68 times faster even though it performs more work. Using a nine-point stencil, it is 3.78 times faster. Values are given for the nested dissection (ND) ordering optimal for a scalar processor (i.e., the iPSC/MX) as well as an incomplete nested discussion (IND) algorithm as would be used on a vector processor (i.e., the iPSC/VX).

A parallel algorithm should be evaluated against the best serial algorithm running on equivalent hardware. Figs. 12 and 13 contrast the DMF algorithm to the serial algorithm from PISCES. Fig. 12 plots run-time versus problem size. Fig. 13 contains the speedups achieved as the problem size increases. The DMF algorithm is run on four, eight, and 16 processors. Scattered data was obtained due to

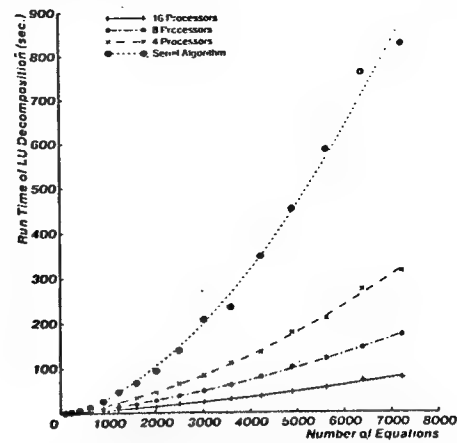


Fig. 12. Run time of sparse matrix factorization as a function of problem size.

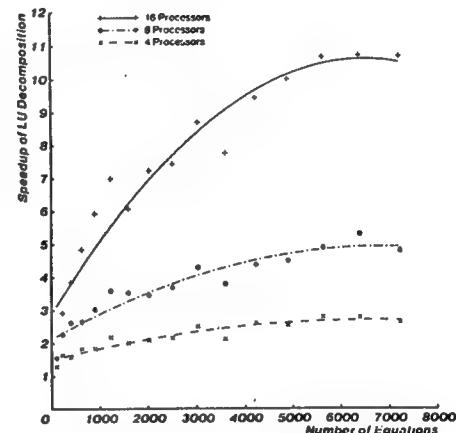


Fig. 13. Speedup achieved by the DMF algorithm as a function of problem size.

TABLE III
AVERAGE NUMBER OF MESSAGES SENT FACTORING 961 EQUATIONS (31×31 GRID)

Stencil	Solution Method and Ordering Strategy				
	Serial Band	Serial MD	GS ¹ ND	DMF ¹ IND	DMF ¹ ND
5 Point	0	0	489	57	57
9 Point	-	-	688	57	57

Stencil	Solution Method and Ordering Strategy				
	Serial Band	Serial MD	GS ¹ ND	DMF ¹ IND	DMF ¹ ND
5 Point	103	27.3	14.66	6.22	5.46
9 Point	-	-	21.66	6.70	5.73

TABLE IV
TIME (SECONDS) SPENT FACTORING 961 EQUATIONS (31×31 GRID)

Solution Method and Ordering Strategy				
Serial Band	Serial MD	GS ¹ ND	DMF ¹ IND	DMF ¹ ND
103	27.3	14.66	6.22	5.46
-	-	21.66	6.70	5.73

¹ Parallel algorithms run on 16 processors
Ordering Strategies:
Band: Minimum Bandwidth
MD: Minimum Degree
IND: Incomplete Nested Dissection
ND: Nested Dissection

variations in the load balance among the processors as well as the quality of the minimum degree ordering of the serial problem. Parallel efficiency is defined as the speedup divided by the number of processors employed.

As the size of the problem increases, the efficiency achieved by the DMF algorithm reaches a value of 67 percent for 16 processors. For eight processors, the maximum efficiency observed was only 61 percent. This is because the nested dissection ordering heuristic was optimized for 16 processors and thus did a poor job of static load balancing when only eight processors were employed. With four processors, the allocation of the grid is optimal and an efficiency of 70 percent was achieved.

The speedup curves in Fig. 13 suggest that the efficiency of the DMF algorithm asymptotically approaches a value of 70 percent. This is contrary to the results in the Appendix which suggest that, as the problem size increases, so should the speedup. There are several factors accounting for the upper limit on efficiency. First, a complete nested dissection of the local blocks was used instead of an incomplete nested dissection. This is the optimal ordering for a nonvector processor as it minimizes the number of arithmetic operations needed to factor the blocks. Unfortunately, it also reduces the growth of the block's share of the arithmetic operations. Furthermore, the resulting blocks are very sparse which increases the run-time overhead of manipulating the sparse data structure. In addition, there is still the communication overhead. Even for large dense matrices, communication will limit efficiency to below 80 percent [15]. Finally, the minimum degree ordering algorithm used by the serial code performs a cyclic reduction of the adjacency graph of the sparse matrix. This permits the serial algorithm to factor the matrix using fewer floating-point operations than the parallel algorithm. Therefore, on one processor, the DMF code is slower than the serial code from PISCES.

Figs. 14 and 15 show the throughput and speedups achieved by the triangular solvers. As expected, the reduced workload and limited concurrency available yields lower speedups. In fact, for the smallest problem, the serial algorithm is faster than the parallel one, even with 16 processors. This observation is easily explained since the parallel implementations required message passing that dominated their run-times. For larger problems, peak speedups of 2.83, 4.14, and 5.71 were achieved with four, eight, and 16 processors. Fortunately, the triangular solvers run much faster than matrix factorization. Therefore, speedup of the entire solution of the matrix equation is virtually unaffected by the relatively poor performance of the triangular solvers. When solving the 75 by 75 system of equations on 16 processors, matrix factorization required 55 s, whereas both FE and BK took only 4.29 s. Overall speedup was 10.26.

VI. CONCLUSIONS

A new distributed multifrontal algorithm for solving large sparse systems of equations has been presented that overcomes the communication bottleneck previously reported for general sparse solvers. An order of magnitude reduction in the communication load of a sample problem has been demonstrated. Using this new technique, parallel processor efficiencies of 70 percent have been ob-

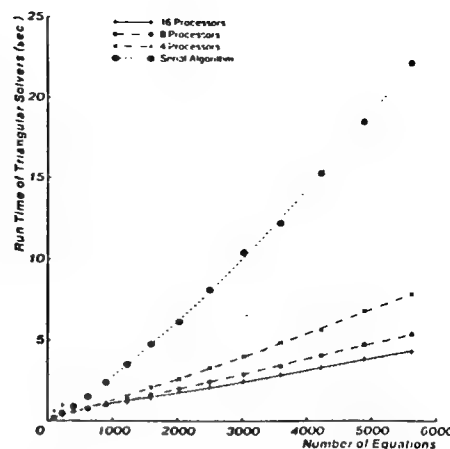


Fig. 14. Cumulative runtime of forward and back solves as a function of problem size.

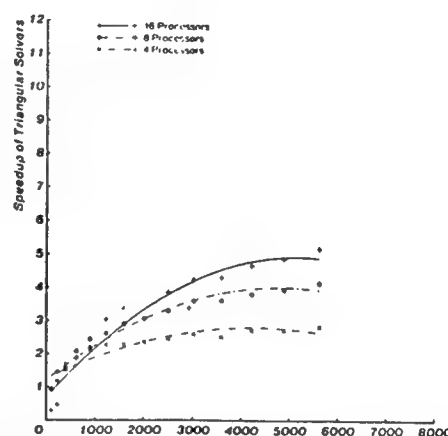


Fig. 15. Speedup achieved by the DMF forward and back solves as a function of problem size.

served. This level of efficiency was observed over both a range of problems and with a varying number of processors. While this algorithm was originally intended for use on a distributed memory hypercube, it should also be applicable to shared memory systems, such as the Cray X-MP4/8. In shared memory systems, the communication overheads manifest themselves as synchronization and mutual exclusion problems.

The communication overhead is minimized by a frontal distribution of physically adjacent pivots' rows and columns to one processor. Separate blocks can be factored without interprocessor communication since updates to their separator fronts are stored locally. Message traffic is also restricted while factoring the separator submatrices. During the dissection process, the blocks of the dissected problem were always divided between logically adjacent processors. Therefore, the set of processors factoring any separator's submatrix is always a complete hypercube of lower dimension (i.e., subcube) embedded within the multiprocessor. All messages needed to resolve data dependencies during the separator factorization are transmitted using a spanning tree that is restricted to the subcube. The messages are limited and remain in the working subcube.

During the triangular solution phase, messages are not constrained to nearest neighbors in the hypercube. These messages must propagate further through the network, which reduces the interprocessor communication rate. For example, during the triangular solves in Fig. 9, messages are exchanged between processors 0 and 3 as well as processors 1 and 2. In addition, the messages are routed through other subcubes of the multiprocessor, thereby interrupting computation that could otherwise be performed independently. These overheads, coupled with the limited potential concurrency available in the FE and BK algorithms, reduced the speedups achieved in the triangular solvers to half those observed for factorization.

Even though this work was motivated by semiconductor device simulation, the sparse matrix solution technique is applicable to a wide range of scientific and engineering disciplines. This work has focused on rectangular grids. However, automatic nested dissection routines can be used to extend the usefulness of the DMF algorithm to problems generated from irregular 2-D grids. Two algorithms for performing nested dissections of general graphs have been published [21], [22] and one implementation is available in Waterloo University's SPARSPAK [23]. Therefore, the distributed multifrontal algorithm is applicable to any sparse matrix problems where the adjacency structure of the matrix can be represented by a planar graph. These include two-dimensional finite-difference and finite-element analysis.

It is not clear how applicable the DMF algorithm will be for matrices derived from nonplanar graphs. Matrices such as those generated in direct methods of circuit simulation can contain coupling between physically remote nodes. Therefore, use of the DMF algorithm in problems such as circuit simulation may require the generation of application-specific nested dissection heuristics and the introduction of restrictions on signal routing.

APPENDIX

The work performed to factor a processor's local block and update its separator front is computed as follows:

Assuming a symmetric LU decomposition, the work at each pivot i will be approximated as twice the square of the number of elements in the pivot column L_i . If the block has separators on two sides, it will look like block 0 in Fig. 5.

Let: m_s be the length of the shorter side of the block
 m_l be the length of the longer side of the block.

$$\begin{aligned} \text{Independent FLOP's} &\approx 2 \sum_{i=1}^{m_s m_l} ((m_s + 1) + ((i/m_s))^2) \\ &\approx 2m_s^3 m_l + 2m_s^2 m_l^2 + (2/3) m_s m_l^3 \\ &\approx (14/3) m^4, \quad \text{if } m_s = m_l. \end{aligned}$$

Similarly, blocks surrounded by three or four separators require the following.

For three separators:

$$\begin{aligned} \text{Independent FLOP's} &\approx 2m_s^3 m_l + 4m_s^2 m_l^2 + (8/3) m_s m_l^3 \\ &\approx (26/3) m^4, \quad \text{if } m_s = m_l. \end{aligned}$$

For four separators:

$$\begin{aligned} \text{Independent FLOP's} &\approx 8m_s^3 m_l + 8m_s^2 m_l^2 + (8/3) m_s m_l^3 \\ &\approx (56/3) m^4, \quad \text{if } m_s = m_l. \end{aligned}$$

It is apparent from these simple equations that the size of the blocks assigned to each protector should vary as a function of the block's location within the problem grid. Otherwise, there will be a serious load imbalance.

ACKNOWLEDGMENT

The authors would like to thank the General Electric Company and Intel Corporation for their generous financial and material support. We would also like to thank C. Moler and D. Scott of Intel Scientific Computers for their assistance in the implementation of our algorithm. Finally, we would like to thank the referees for the improvements they suggested.

REFERENCES

- [1] J. A. George *et al.*, "Sparse Cholesky factorization on a local memory multiprocessor," Tech. Rep. ORNL/TM-6190, Mathematical Sciences Section, Oak Ridge National Lab., Oak Ridge, TN 37831, 1985.
- [2] M. R. Pinto, C. S. Rafferty, and R. W. Dutton, "PISCES-II: Poisson and continuity equation solver," Stanford Electronics Lab. Tech. Rep., Sept. 1984.
- [3] C. S. Rafferty and M. R. Pinto, private conversation, Elect. Eng. Dep., Stanford Univ., June 1984.
- [4] G. Jacob *et al.*, "Direct-method circuit simulation using multiprocessors," in *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 170-174, 1986.
- [5] I. S. Duff and J. K. Reid, "The multifrontal solution of indefinite sparse symmetric linear equations," *ACM Trans. Math. Software*, vol. 9, no. 3, pp. 302-325, Sept. 1983.
- [6] G. A. Giest and M. T. Heath, "Parallel Cholesky factorization on a hypercube multiprocessor," Tech. Rep. ORNL/TM-9962, Mathematical Sciences Section, Oak Ridge National Lab., Oak Ridge, TN 37831, 1985.
- [7] G. Alghband and H. F. Jordan, "Multiprocessor sparse L/U decomposition with controlled fill-in," Tech. Rep. 85-48, ICASE, NASA Langley Research Center, Hampton, VA 23665, 1985.
- [8] H. T. Kung and C. E. Leiserson, *Algorithms for VLSI Processor Arrays*. Reading, MA: Addison-Wesley, 1980, pp. 271-292.
- [9] C. P. Arnold, M. I. Parr, and M. B. Dewe, "An efficient parallel algorithm for the solution of large sparse linear matrix equations," *IEEE Trans. Computers*, vol. C-32, Mar. 1983.
- [10] S. Lundstrom, "A decentralized control, highly concurrent multiprocessor," in *Proc. 1985 Computer Architecture Symp.*, 1985.
- [11] S. Baden, private conversation, EECS Dep., Univ. California, Berkeley, Aug. 1986.
- [12] J. L. Larson, "Multitasking on the Cray X-MP-2 multiprocessor," *Computer*, vol. 17, no. 2, pp. 62-69, July 1984.
- [13] J. A. George and J. Liu, *Computer Solution of Large Sparse Positive Definite Systems*. Englewood Cliffs, NJ: Prentice Hall, 1981.
- [14] G. Golub and C. Van Loan, *Matrix Computations*. Baltimore, MD: The Johns Hopkins University Press, 1983.
- [15] C. Moler, First conference on hypercube multiprocessors, Knoxville, TN, Aug. 1985.
- [16] O. Wing and J. W. Huang, "A computation model of parallel solution of linear equations," *IEEE Trans. Computers*, vol. C-29, pp. 632-638, July 1980.
- [17] M. A. Srinivas, "Optimal parallel scheduling of Gaussian elimination DAG's," *IEEE Trans. Computers*, vol. C-32, pp. 1109-1117, Dec. 1983.

- [18] B. M. Irons, "A frontal solution program for finite element analysis," *Int. J. Numerical Methods Eng.*, vol. 2, pp. 5-32, 1970.
- [19] J. A. George, J. Liu, and E. Ng, "Communication reduction in parallel sparse Cholesky factorization on a hypercube," 2nd Conf. on Hypercube Multiprocessors, Knoxville, TN, Sept. 1986.
- [20] J. A. George, *et al.*, "Incomplete nested dissection for solving n by n grid problems," *SIAM J. Numerical Analysis*, vol. 15, no. 4, pp. 662-673, Aug. 1978.
- [21] J. A. George and J. Liu, "An automatic nested dissection algorithm for irregular finite element problems," *SIAM J. Numerical Analysis*, vol. 15, no. 5, pp. 1053-1069, Oct. 1978.
- [22] R. J. Lipton and R. E. Tarjan, "Generalized nested dissection," *SIAM J. Numerical Analysis*, vol. 16, no. 2, pp. 346-358, Apr. 1979.
- [23] E. Chu *et al.*, "User's Guide for SPARSPAK-A," Res. Rep. CS-84-36, Dep. of Computer Science, Waterloo Univ., Waterloo, Ontario.

*



Robert F. Lucas was born in Detroit, MI, on Aug. 1, 1958. He received the B.S.E.E. and M.S.E.E. degrees from Stanford University, Stanford, CA, in 1980 and 1983, respectively.

From 1980 to 1984, he worked for Hughes Aircraft Co., Fullerton, CA. He has also held summer positions at Hughes, in 1979, at the General Electric Company's Corporate Research and Development Center, Schenectady, NY, in 1984, and with Intel Scientific Computers, Beaverton, OR, in 1986. He is currently working toward the Ph.D.

degree in electrical engineering at Stanford University. His research interests are in parallel processing architectures and algorithms.



Tom Blank received the B.S.E.E. degree from the University of Washington and the Ph.D. degree in electrical engineering from Stanford. While attending Stanford, he also worked at Hewlett-Packard.

After finishing his degree, Dr. Blank worked as a Stanford research associate, then as an independent consultant before returning to Stanford in his current position as an Acting Assistant Professor. His research interests are in high-level electrical CAD, specifically parallel algorithms, high-level synthesis, hardware accelerators, and incremental design tools.

*



Jerome F. Tiemann (M'59-SM'65-F'76) was born February 21, 1932. He received the Sc.B. degree from the Massachusetts Institute of Technology in 1953 and the Ph.D. degree from Stanford University in 1960.

Dr. Tiemann has been employed by the General Electric Corp. at the Corporate Research and Development Center since 1957 where he has worked on a variety of projects including electronic phenomena in semiconductors, circuit techniques for solid-state devices, and system architectures for medical diagnostic imaging, communication, television, sonar, and radar. His interest in computational simulation of semiconductor devices dates from 1960, when he started development work on Tunnel diodes. Since that time, he has been an advocate of predictive simulation for all aspects of circuit and system design. As a result of receiving a Coolidge Fellowship from his Laboratory, he spent a Sabbatical year at Stanford University during 1984 and 1985, where he pursued his interests in computational aspects of the simulation of physical phenomena for engineering purposes.

Dr. Tiemann is a member of APS and the National Academy of Engineering.

A STRIDE Towards Practical 3-D Device Simulation—Numerical and Visualization Considerations

Ke-Chih Wu, *Member, IEEE*, Goodwin R. Chin, *Student Member, IEEE*, and Robert W. Dutton, *Fellow, IEEE*

Abstract—A 3-D device solver (STRIDE), capable of solving grids up to 250 000 nodes, has been developed on a message passing multiprocessor. By the use of iterative matrix solvers and Gummel style nonlinear iteration schemes, user memory per node is reduced over use of direct solvers and Newton schemes. By using an independent-edge-grouping scheme to increase the vector length to the order of the number of variables, the vector processing efficiency is significantly increased without additional floating point operations. We extend the modified-singular-perturbation (MSP) scheme to two-carrier simulations. This significantly speeds up the convergence rate of Gummel style nonlinear iterations. Physical insight gained from the MSP schemes also leads to an automatic switching scheme between various nonlinear schemes based on the monitoring of certain matrix parameters. This allows the incorporation of a previously proposed Newton-1C scheme which offers the best CPU performance for normal bipolar simulations. When combined with current convergence criterion, a set of MSP inspired convergence criterion are better able to recognize a practically converged solution. A novel global convergence scheme is also developed based on insight from MSP principles. Interactive user interface and links to graphics tools are provided to support the tool integration efforts. Application of STRIDE is demonstrated by an analysis of latchup trigger current dependence on layout arrangement.

I. INTRODUCTION

WITH THE continuing miniaturization of integrated circuits, 3-D effects significantly impact device characteristics. A robust and efficient 3-D device solver will give device engineers significant leverage in pursuing state-of-the-art IC technologies. Various 3-D simulators (e.g., [1]–[4]) have appeared to address these needs. However, one of the major hurdles which has prevented widespread use of 3-D device simulation is the vast amount of computational resources required for such an endeavor, as the number of variables can easily run into hundreds of thousands, or even millions. Multiprocessors, which connect together a large number of inexpensive processors, provide a cost-effective platform for CPU-intensive 3-D simulations. To explore the potential

for a cost effective 3-D device simulator, we have developed STRIDE (Stanford ThRee dImensional Device simulator) on a message passing multiprocessor (Intel iPSC2). This paper describes the progress that has been made since the previous report [5], and mainly centers around the various computational aspects with special emphasis on bipolar simulations. Our experience in 3-D visualization is also discussed.

Section II gives an overview of the device solver. Section III discusses schemes which increase the vector length to the order of number of variables for the sparse matrices encountered in 3-D simulation. In Section IV, various modified singular perturbation (MSP) schemes are introduced for two-carrier simulations which significantly improve the convergence of Gummel style nonlinear iterations. The results of a previously proposed Newton-1C scheme will be presented which offers the best CPU performance with less memory than the full-Newton scheme. A MSP inspired matrix parameter will be introduced which allows a switching scheme that automatically chooses the best nonlinear scheme for the simulation. In Section V, other applications of MSP principles will be discussed which include a new set of convergence criterion capable of determining practically converged solutions and a novel global convergence scheme. Section VI discusses our approaches in developing better user interfaces and on tool integration aspects. Section VII, presents an application example of STRIDE in the analysis of the latchup trigger current's dependence on electrode arrangement. Finally, conclusions are drawn in Section VIII.

II. OVERVIEW OF STRIDE

In STRIDE, up to two current continuity equations are solved together with Poisson's equation. In normalized form, these equations are given by

$$\nabla \cdot (\epsilon \nabla \psi) = n - p + N_A - N_D \quad (1)$$

$$\nabla \cdot J_n - U = 0 \quad (2)$$

$$\nabla \cdot J_p + U = 0 \quad (3)$$

where $n = n_{ie} \exp(\psi - \phi_n)$, $p = n_{ie} \exp(\phi_p - \psi)$, $J_n = -q\mu_n n \nabla \phi_n$, and $J_p = q\mu_p p \nabla \phi_p$. The normalization constants used to obtain (1)–(3) are: thermal voltage (kT/q)

Manuscript received July 3, 1990. This work was supported by the Intel Corporation, Texas Instruments, and the U.S. Army Research Office under Contract DAAL03-87-K-0077. This paper was recommended by Guest Editor M. Law.

The authors are with the Integrated Circuits Laboratory, AEL 231 F, Stanford University, Stanford, CA 94305.

IEEE Log Number 9101058.

for electrostatic potential ψ and quasi-Fermi levels ϕ_n and ϕ_p , intrinsic carrier concentration n_i for carrier and impurity concentrations, and the intrinsic Debye length $\sqrt{\epsilon kT/q^2 n_i}$. Effective intrinsic carrier concentration n_{ie} is obtained using the Slotboom bandgap narrowing model [6]. Boltzmann statistics are assumed as can be seen in the formula for n and p . Tabulated doping-dependent mobility values are used. Tangential field dependent mobility is implemented with the Caughey-Thomas model [7].

The main development vehicle for STRIDE has been a message passing hypercube—the Intel iPSC2. The advantages of the hypercube architecture are that it scales to massively parallel systems and that the diameter of the system (average communication delay between the processors) grows only logarithmically with the number of processors. An important feature of the Intel hypercube is the amount of the memory per processor. The system used in this work has 16 processors, each with 8 Mbytes of memory. The sustainable performance for the system is about 1.5 MFlops [8] in which each processor constitutes an Intel 386 paired with a 387 math coprocessor. The development has recently been shifted onto an iPSC/860 system which has 32 processors each with 16 Mbytes of memory. Preliminary results have shown a system performance of approaching 100 MFlops. STRIDE also runs on Convex C1 and Cray YMP.

The simulation domain is currently approximated by a 3-D rectangular grid with provisions for nonplanar structure [5]. Work is going on to develop parallel algorithms for dealing with general grids generated by grid generators such as OMEGA [9]. Equations (1)–(3) are discretized using the finite difference method. In discretizing the continuity equations, Scharfetter-Gummel current formulation [10] is used.

The discretization of (1)–(3) yields a nonlinear system of algebraic equations which are solved by one of several nonlinear iteration schemes implemented in STRIDE. For each nonlinear iteration in Gummel's scheme, the discretized Poisson's equations ($F_\psi(\psi, \Phi_n, \Phi_p) = 0$) are solved for the update vector $\delta\psi$ holding Φ_n and Φ_p constant.¹ This is achieved by repeatedly solving

$$A_{\psi\psi}\delta\psi = -F_\psi \quad (4)$$

given the current estimate of ψ , Φ_n and Φ_p . In (4), $A_{\psi\psi} \equiv (\partial F_\psi(\psi, \Phi_n, \Phi_p)/\partial\psi)$ is called the main matrix of Poisson's equation. Other matrices are similarly defined. The discretized current continuity equations ($F_{\Phi_n}(\psi, \Phi_n, \Phi_p) = 0$ and $F_{\Phi_p}(\psi, \Phi_n, \Phi_p) = 0$) are then solved. Since $A_{\Phi_n\psi}$ and $A_{\Phi_p\psi}$ are linear, one matrix solution will suffice. This process repeats until convergence is achieved. For other nonlinear schemes, two of the equations are solved together while the other is solved separately. More details of these schemes are discussed in Section IV. The convergence criteria are the maximum magnitude of ψ updates, terminal current conservation, and relative change in the

magnitude of terminal currents and of terminal charges. Further discussions are deferred to Section V.

The matrix solutions are the most CPU time intensive steps in STRIDE. The incomplete Cholesky conjugate gradient (ICCG) algorithm [11] is used to solve the symmetric matrices, while asymmetric matrices are solved using the incomplete LU decomposition conjugate gradient squared (ILUCGS) algorithm [12]. The parallel implementation of these algorithms, which are based on domain decomposition, are described in [13] and [14]. The parallel efficiency achieved by these algorithms, while running on 16 processors, is more than 80%² when the problem size exceeds 50 000 nodes.

The maximum number of grid points that can be handled by STRIDE on the 16-node iPSC/2 system is over 100 000,³ which translates to a cubic grid of 47 points in each dimension. This is the direct result of not using the full-Newton scheme which would nearly double the memory per node. CPU time per bias point is about 1.5 h for a 70K node bipolar example. This is averaged from a I_c versus V_{be} curve with $V_{ce} = 5$ V. In this curve, V_{be} increases from 0.4 to 1 V in 0.1-V steps.

III. VECTORIZATION SCHEMES

Vectorization is an important aspect of reducing the execution time of the program. Since a majority of CPU time is spent solving matrices, our efforts have concentrated on vectorizing the iterative matrix solvers.

The principle behind the vectorization is to group together long chains of repetitive operations which are mutually independent. This independence is essential so that vector processing will not produce different results from the scalar operations. Thus the key to vectorization is to identify such groups of operations. For most iterative matrix solution algorithms, most of the operations involve vector-vector or matrix-vector products. Although the former is trivially vectorized, the latter takes some effort when the matrices are sparse. A matrix-vector product can be considered to be the sum of many vector-vector products which can be easily vectorized. This works well for dense matrices which have long rows. However, when the matrix is sparse, the length of these vectors becomes very short (typically, three to six) which seriously impedes vector processing performance.

One approach to increase the vector length is to split the matrices into many small dense matrices obtained from the elements of the simulation domain, such as triangle elements in 2-D simulation [15]. When two elements contain no common node, their matrices are independent and can be grouped together. This grouping can be called independent element grouping.

Building upon this idea, we implemented an independent edge grouping scheme. In terms of group theory, a

¹ Φ_n and Φ_p indicates the use of Slotboom variable in the continuity equation.

²Previously, we have reported a parallel efficiency of about 60% when the concurrent ICCG algorithm ran on iPSC. The improvement in efficiency is a result of the ten-fold improvement in the data latency for iPSC/2 than iPSC.

³The maximum grid count is increased to more than 250 000 on the new 32-node iPSC/860 system with 16 Mbytes per node.

matrix element (A_{ij}) can be considered as an edge between the row node (i) and the column node (j). When two edges contain no common node, the matrix-vector operations they represent are independent and can be grouped together. Due to the above restriction, there can be only one edge that contains a particular node in each group. Thus the minimum number of groups is the maximum number of edges a node has. For a seven-point stencil, this number is six (the diagonal elements are already grouped together in the sparse matrix data structure). The grouping is achieved by a greedy algorithm searching through the edges represented in the sparse matrix pointers. So far, optimal grouping, in the sense that six groups are sufficient to cover all the edges, has always been achieved with our present ordering schemes of the nodes. The average size of the groups, which equals the average vector length, is about half the number of nodes.

When compared with the element grouping, the edge grouping has the advantage of not requiring extra double precision storage and extra floating point operations. Furthermore, it is compatible with the parallel implementation of the matrix solver on the hypercube [14]. The disadvantage is that more indirect addressing is needed which slows down the vector operations. This disadvantage is partially alleviated by re-ordering the matrix elements so that the indirect addressing is not needed to access the matrix. This change has resulted in a 25% increase in CPU performance for Cray YMP, which is also the average improvement seen on Convex C1.

With the matrix-vector product vectorization issue resolved by the above mentioned schemes, all the operations in the conjugate gradient algorithm are now well vectorized. Extra efforts are needed to vectorize the operations involving the IC or ILU preconditioner which account for more than one third of the total operation counts. A well-known scheme is to color order the nodes. Coloring divides the nodes into several groups such that a node will not be in the same group with any nodes that share an edge with it. For the seven-point stencil finite-difference scheme currently used in STRIDE, only two colors are necessary and the ordering scheme is called red-black ordering. The price for the red-black ordering is an increase in iteration number. From our experience, the increase in the iteration numbers is about 30% for symmetric matrices derived from uniform grids and about double for asymmetric matrices. Still the advantages stemming from the ability to fully vectorize the entire matrix solver operation outweighs its penalties. The performance of the algorithms were measured in terms of CPU time per linear iteration. When measured in terms of CPU time per linear iteration per variable, the raw speed advantage of vector over scalar for ICCG and ILUCGS type of iterative matrix solvers is about 4.5 using the Convex C1 and 9.5 using the Cray YMP.⁴ Therefore, even with

the worst-case situation, red-black ordering reduces the computation time of ICCG and ILUCGS operations by more than 50% on Convex C1 and more than 80% on Cray YMP.

When implemented on Convex C1 and Cray YMP, the iterative matrix solvers in STRIDE are able to run at 2 MFlops on the C1 and 100 MFlops on the YMP.

IV. ACCELERATION OF TWO-CARRIER GUMMEL STYLE ITERATIONS

Having achieved dramatic improvement in the convergence performance of Gummel style nonlinear scheme at high level injection using a MSP scheme [5], our attention turned to the application of MSP and its extensions to Gummel style iterations in two-carrier simulation. We will call the MSP scheme proposed in [5] MSP-1C, with 1C added for one-carrier.

For completeness, the key formula for MSP-1C is shown in the following:

$$D_{\psi\psi}\delta\bar{\psi} + D_{\psi\phi_n}\delta\bar{\phi}_n = -F_{\psi}. \quad (5)$$

The key point from the discussion of MSP-1C [5] is that in the n -type region where the charge neutrality prevails, (5) is quite accurate and its substitution into the linearized continuity equation will retain much of the coupling between Poisson and continuity equations, thereby improving the convergence performance of the Gummel style nonlinear iteration scheme.

Two simple extensions of the MSP-1C scheme, which retain the advantage of low computational cost per iteration, can be used in two-carrier simulations. One is to apply MSP-1C to the "main" carrier equation, such as the electron continuity equation in n-p-n transistor simulations. The other is to use MSP-1C separately on each continuity equation. The advantage of these extensions is low computational cost per iteration. For both cases, the presence of the other carrier is ignored as far as MSP-1C is concerned. Therefore, dramatic improvement in convergence performance is not to be expected. Nevertheless, significant improvement has been observed over the traditional Gummel's scheme with the asymptotic convergence rate for these schemes ranging from four to six of that for the Gummel iteration in the high-level injection regime. However, these increasing convergence rates are still very low with the error typically halving every six to seven iterations.

These unsatisfactory results prompted us to explore new schemes. Our first approach was to use a "true" extension of MSP-1C, the MSP-2C scheme. The key formula for this MSP-2C scheme is shown as follows:

$$D_{\psi\psi}\delta\bar{\psi} + D_{\psi\phi_n}\delta\bar{\phi}_n + D_{\psi\phi_p}\delta\bar{\phi}_p = -F_{\psi}. \quad (6)$$

Comparing (5) with (6), the terms associated with changes in both carrier variables are included, thereby the name MSP-2C. When (6) is substituted into the linearized continuity equations of both carriers, we obtain a matrix with a dimension of $2N$ by $2N$. This matrix can be expressed in terms of the original matrices as follows:

⁴The vectorization on iPSC/2 was not pursued because of the design flaw in vector processing unit (VPU). As it stands, a VPU can only access about one eighth of the total memory and a complete vectorization of the iterative solvers would entail constant data swapping. The newly available i860-based systems does not have such a problem.

$$A_{\text{MSP-2C}} = \begin{bmatrix} -A_{\psi n} D_{\psi\psi}^{-1} & I & 0 \\ -A_{\psi p} D_{\psi\psi}^{-1} & 0 & I \end{bmatrix} \begin{bmatrix} D_{\psi n} & D_{\psi p} \\ A_{\psi n} & D_{\psi n} \\ D_{\psi p} & A_{\psi p} \end{bmatrix}. \quad (7)$$

As the size of $A_{\text{MSP-2C}}$ indicates, the two continuity equations are solved together with the MSP-2C while Poisson's equation is still solved separately. Solution of larger matrices results in an increase in both the data storage and CPU time per iteration. This is the major factor that reduces the maximum simulatable node count from that previously reported to 130 K to about 100 K. CPU time per iteration has been observed to roughly double.

Somewhat to our surprise, the incorporation of MSP-2C does little to improve the convergence of normal transistor simulations beyond what has been achieved by the MSP-1C scheme, although in the latch-up analysis, the convergence behavior of the latched device improves dramatically with the error at least halving for every iteration.

In order to stay within the memory requirement of MSP-2C scheme instead of going full Newton and a double of memory requirement, we next turned an algorithm which solves Poisson equation with the "main" carrier equation such as the electron equation in a n-p-n transistor, the Newton-1C scheme.⁵ An additional motivation for using the Newton-1C scheme was the observation that throughout the simulation of normal bipolar transistor operation, the coupling between Poisson and the "minor" carrier equation remained very weak⁶ even though the device itself had gone into the strong high-level injection regime.

Fig. 1 shows the convergence results for Gummel, MSP-1C and Newton-1C schemes for simulations done on a bipolar transistor. V_{CE} is fixed at 5 V. The number of nodes is about 13 700 and the simulations are executed on eight processors with an estimated parallel efficiency of 72%. As shown in Fig. 1, at the highest injection level, MSP-1C is about three times faster than Gummel, while Newton-1C is still three times faster than MSP-1C, despite the doubling in CPU time per iteration. Although the full Newton scheme is not yet available from STRIDE, Newton-1C is expected to be faster than the full Newton scheme since CPU time per iteration for the full Newton is expected to be twice of that for Newton-1C. For instance, for the test example in the next section, a total of eighteen iterations are needed for convergence, which is CPU equivalent to less than eight full Newton iterations. Given the severity of the test example, it is very unlikely that the full Newton scheme can converge in less than eight iterations.

It should be noted that the kind of matrix solvers used in a device solver affects the results obtained for using the

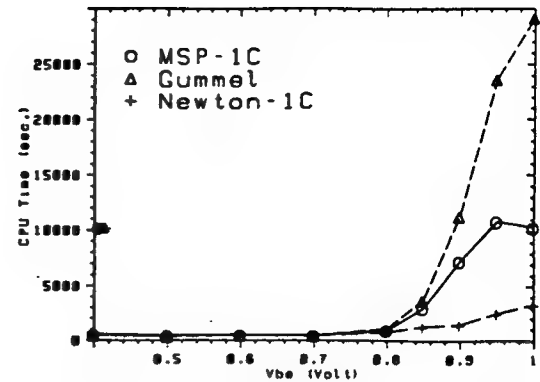


Fig. 1. Two-carrier convergence results of various schemes.

MSP and Newton-1C schemes. When a device solver uses iterative device solvers, forward elimination and backsubstitution as well as matrix-vector multiplication are the most CPU intensive operations. The cost of these operations grows as the *square* of the number of variables per node. In a device solver using a direct matrix solver, however, the cost of matrix factorization usually dominates CPU time and it grows as the *cube* of the number of variables per node. Therefore, the proper use of the MSP and Newton-1C schemes in such a device solver are expected to yield an even more favorable result in comparison with the use of the full Newton scheme.

Given the results of these schemes, one might ponder the reason behind the relative success of MSP in one-carrier simulation and its relative ineffectiveness in two-carrier simulations even though the minor carrier equation is only weakly coupled to the Poisson equation. The key formula of the MSP scheme, (5) and (6), relates the relevant carrier concentration with the value of ψ at the high carrier concentration nodes. Therefore, MSP performs best when the high-level injection only causes the local coupling between Poisson and continuity equation(s), that is, when the value of ψ at a node is the dominant factor in determining the carrier concentration at that node. This is the situation in the inversion layer of MOSFET's where the conduction charge is induced electrostatically. The situation for the two-carrier simulation is very different. High-level injection of a bipolar transistor almost always accompanies the Kirk effect [16], i.e., the base push-out effect. When the Kirk effect occurs, the carrier concentration in the lightly doped collector region is determined not by the local values of ψ , but rather the amount of the collector current that needs to be sustained. This is in turn determined by the injection level at the base-emitter junction. On the other hand, the amount of carrier concentration also significantly impacts the ψ distribution in the lightly doped collector region. Therefore, a nonlocal coupling exists between the Poisson and continuity equations. Since MSP schemes are only capable to take into account the local coupling between Poisson and continuity equation(s), they are relatively ineffective in improving the convergence of Gummel style iterations in normal bipolar simulations. On the other hand, since it takes into account the complete coupling between Poisson and the

⁵The Newton-1C scheme was used in some early works on device simulation, such as an early version MINIMOS and Dr. J. W. Slotboom's initial work on 2-D simulation some 15 a ago.

⁶This can be ascertained by noticing that the error of the other continuity equation is several orders below that of the main continuity equation.

main carrier equation by solving them together, the Newton-1C scheme is able to achieve full-Newton competitive convergence performances, given that the minor carrier equation is virtually independent of the Poisson equation.

The fact that Newton-1C is more expensive than MSP-1C in CPU time per iteration presents an issue of when to switch from Gummel or MSP-1C schemes to Newton-1C in order to optimize the overall solution time. During the implementation of MSP-1C scheme, we found an interesting parameter which has important bearing on this issue. Excluding the recombination terms, the discrete current continuity equation at node i consists of the sum of all the current components flowing into the node. For an electron current component which flows into node i along the edge connecting the nodes i and j ($I_{n,ij}$), there are contributions to diagonal $A_{\Phi_n\Phi_n}(ii)$ ($\partial I_{n,ij}/\partial \Phi_n(i)$) and off-diagonal $A_{\Phi_n\Phi_n}(ij)$ ($\partial I_{n,ij}/\partial \Phi_n(j)$). From the point of view of matrix formulation, the application of MSP-1C scheme means a modification to the original matrix of the current equation (i.e., $A_{\Phi_n\Phi_n}$ is modified by $-A_{\Phi_n\psi} \cdot DD_{\psi\psi}^{-1} D_{\psi\Phi_n}$ for electron equation [5]). For $I_{n,ij}$, the modification to $A_{\Phi_n\Phi_n}(ii)$ is $-(\partial I_{n,ij}/\partial \psi(i)) D_{\psi\psi}^{-1}(i) D_{\psi\Phi_n}(i)$. A detailed analysis reveals that, if $\Phi_n(i) < \Phi_n(j)$, then the modification strengthens $A_{\Phi_n\Phi_n}(ii)$, i.e., it has the same sign with $A_{\Phi_n\Phi_n}(ii)$; otherwise, $A_{\Phi_n\Phi_n}(ii)$ is weakened by the modification. This diagonal weakening merits special attention, since, according to our experience, all the diagonals in a matrix must have the same sign for the iterative matrix solvers to converge. It turns out that the ratio ($\beta_{n,ij}$) between the modification $(-\partial I_{n,ij}/\partial \psi(i)) D_{\psi\psi}^{-1}(i) D_{\psi\Phi_n}(i)$ and the original term $(\partial I_{n,ij}/\partial \Phi_n(i))$ is never less than minus one. This ensures the diagonals of $A_{\Phi_n\Phi_n}$ will not become negative as the result of MSP modifications. Furthermore, the most negative of all the $\beta_{n,ij}$ can be used as a parameter, indicating the degree to which the original $A_{\Phi_n\Phi_n}$ has been modified.⁷ Our experience indicates that when the magnitude of this parameter is small, the Gummel iteration is just as effective as any other nonlinear scheme. When the magnitude approaches to one, however, the Gummel iteration becomes very slow and a more elaborate scheme has to be employed to accelerate the convergence. Therefore, this parameter is also a very good indication of the strength of the coupling between Poisson's equation and the continuity equation. A scheme is thereby implemented in STRIDE to use Newton-1C, if the user desires to do so, when the magnitude of this matrix parameter is large enough. Currently, the threshold value is 0.25. The low bias results of Newton-1C curve in Fig. 1 were actually obtained with Gummel or MSP-1C schemes. By the same token, this switch scheme can also be extended to use the full Newton scheme when it is truly necessary. In short, the introduction of this matrix parameter makes it feasible for the program to automatically choose the best nonlinear scheme at its disposal. Although originated in the context

of the finite difference approach, this parameter can also be calculated in device solvers using the finite element approach with only minimum overhead.

V. CONVERGENCE CRITERION AND DAMPING SCHEMES

Traditionally, the convergence criterion for the carrier variables have been that the maximum relative change of all the variables is below a certain value.⁸ In contrast to this relative criterion, there is also the absolute criterion which measures the convergence of residual of the difference equations. Although the absolute criterion is useful in a global damping scheme [17], its application as a convergence criterion is less useful. For example, the residual of the current continuity equation which can be tolerated depends heavily on the amount of current flowing through the simulating device, which can differ by many orders of magnitude. It is, therefore, not always possible to determine *a priori* what is the appropriate tolerance level for the residual. Another important criteria that has not been widely used is the convergence of the terminal currents, which also entails the conservation of all the terminal currents. When we monitor the convergence of both the currents and the variables, we found that in the low current regime, the variables may converge before the currents do. Instances were observed in which the currents do not converge until the maximum relative change in carrier variable fell below 10^{-9} or even lower. By combining these two convergence criteria, we are able to reduce the lower limit above which the currents can be calculated self-consistently. On the other hand, at high current levels, variables may lag far behind the currents in convergence. In this regard, we observed instances in which the convergence of current was achieved before the maximum relative change in the carrier variable reached below 10^{-2} . We feel that it is of not much use to calculate the variables to five or six digits of accuracy when the currents have converged. Based on the previous observations, we have chosen a combination of a relatively loose variable convergence criterion about 10^{-2} with a current convergence criterion of about 10^{-4} which has worked quite well for us so far regardless of current levels.

A more serious issue is that for a device simulator using iterative matrix solvers, the absolute convergence of carrier variables becomes much more difficult since matrices are usually solved to a less accurate extent than direct solvers are. In fact, we found that when the traditional Slotboom variables are used, the convergence of these variables become almost impossible in typical two-carrier simulations. When the scaled Slotboom variables are used, however, more often than not, this apparent non-convergence of the concentration variables does not impede the convergence of current which is also accompanied by the convergence in the errors in the continuity equations. Therefore, we have encountered situations in

⁷Although an electron equation is used as the example, a similar parameter can also be calculated for the hole equation.

⁸The convergence criteria for ψ usually measures its maximum absolute change since it is well scaled ($\delta\psi \ll 1$ means the solution is very close.)

which the solutions have converged for all practical purposes, but were not recognized by the traditional convergence criterion for the concentration variables. Therefore, the question is whether we can find a better criterion which can tell when a nonconvergence of carrier variables is for real. The clue again lies in (5) and (6). In the first-order approximation, the terms like $-D_{\psi\psi}^{-1}D_{\psi\phi_n}$ and $-D_{\psi\psi}^{-1} \cdot D_{\psi\phi_p}$ represent the changes in ψ necessary to restore the charge balance required by Poisson's equation. If these changes are very small after the solution of carrier variables, then there are virtually no changes necessary in ψ and we have a converged solution. A detailed look at these terms reveals that they are the relative changes in carrier variables multiplied by a weighting vector whose values are no larger than unity. Therefore, the convergence criterion for the carrier variables can be these weighted relative changes. The weighting coefficients are very small for the minority carrier, and approach one for the majority carrier in the charge neutral region. In essence, this criterion discounts the changes in minority carrier concentration as long as these changes do not significantly perturb Poisson's equation nor impede the convergence of terminal currents. There is a very good correlation between the new measures and the maximum error in the continuity equations and we are able to distinguish the practically converged solutions from the apparently non-converging solutions according to the old measures. The data storage and computational cost of the new measures are minimum since only vector (not matrix) operations are involved.

One of the challenging issues of the nonlinear iteration schemes is how to choose a robust damping scheme to ensure global convergence. The schemes used in STRIDE for Gummel and MSP schemes have been discussed in [5]. For the MSP-2C scheme, both $D_{\psi\psi}^{-1}D_{\psi\phi_n}$ and $D_{\psi\psi}^{-1}D_{\psi\phi_p}$ are considered and different limiting values are used. The problem for the Newton-1C scheme is more difficult since we have two variables with vastly different ranges and a change in ψ impacts the carrier concentration exponentially. Our first attempt was to try various residual limiting schemes such as suggested in [17]. Applications based on their norm reduction principles have proven to be quite successful in nonlinear iteration of Poisson's equation [5]. The results have not been consistent, however. The key difficulty here is how to weigh the residuals from the Poisson and continuity equations.

Our recent attempts are based on a different principle. Since ψ impacts carrier concentration exponentially, the matrix will change dramatically for a large change in ψ . Therefore, the changes in ψ should be restricted so as not to cause large changes in carrier concentration which would greatly upset the charge balance in the previous solution. Similarly, the changes in the carrier variable should be restricted to require only a modest change in ψ to restore the charge balance. In essence, our scheme is a trusted region approach, widely used in the nonlinear optimization community, with the trusted region determined based on the specific knowledge of semiconductor equa-

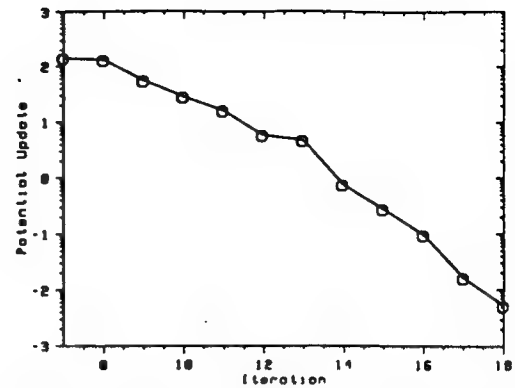


Fig. 2. Global damping scheme: evolution of potential update.

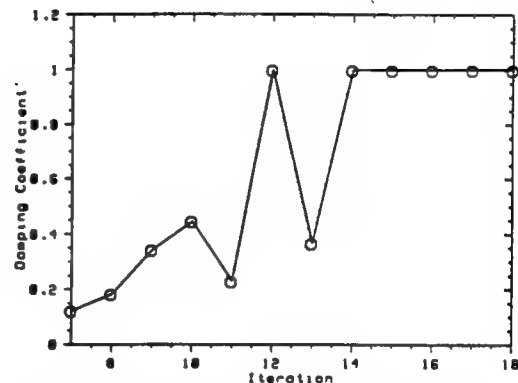


Fig. 3. Global damping scheme: evolution of alpha.

tions. In the actual implementation, a damping coefficient α is first calculated based upon the above principles, and the errors in Poisson and continuity equation are re-evaluated after the damped variable update to monitor the change in these errors. The use of a damped update on the first try prevents the problem of machine overflow which may occur with a full update when $\delta\psi$ become too large. It is rare that a such calculated α still produces an update which causes a dramatic increase in these errors. When this occurs, α is further reduced to brighten down these increases. Our experience shows that it is essential not to insist on the reduction of residuals as long as their increase is moderate. Otherwise, excessive damping may occur which slows the convergence process to a crawl. Fig. 2 shows the evolution of the magnitude of ψ update generated by the Newton-1C scheme during a n-p-n bipolar transistor simulation for an initial bias of $V_{CE} = 5$ V and $V_{BE} = 1$ V. This is a very severe test example for the global damping scheme in that a bipolar transistor is biased into heavy high-level injection regime in one step. Fig. 3 shows the values of the damping coefficient α used, while Fig. 4 shows the errors in the Poisson and continuity equations. Both errors are normalized by their respective starting errors at iteration 7. The number of iteration starts at 7 since Newton-1C was first engaged at this iteration after solution was settled down somewhat. Although the initial (normalized) ψ update is more than 100, which translated to about 3 V, it goes down to less than one (about 26 mV) after just seven iterations. After-

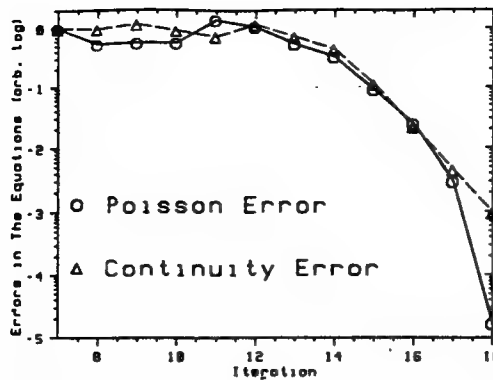


Fig. 4. Global damping scheme: evolution of the residuals.

wards, the solution converges superlinearly as can be seen from the evolution of Poisson errors.

VI. USER INTERFACES AND TOOL INTEGRATION

To provide easier use of STRIDE, we have created a utility program that encapsulates the details of the program and its algorithms from the user. The program, based on the PISCES-IIB utility, BIPMESH, provides automated input deck generation, which includes mesh, model, and bias information. The program asks the user questions about structure geometry, doping profiles, and bias conditions. From this input, mesh is automatically generated using heuristics based on our experience with PISCES-IIB simulation. Doping profiles may be either analytic functions, SUPREM-IV profiles, or SUPREM-III profiles. The extension in the other dimension(s) for SUPREM-IV(III) profiles is(are) based on an error function approximations in the other dimension(s).

The use of automated input deck generation facilitates integration of STRIDE into an integrated simulation system for TCAD. STRIDE will be included as part of a suite of device simulation tools in an integrated simulation environment based upon SIMPL-IPX [18].

To help the user interpret the output of STRIDE, a link to Visualization tools such as NCSA XImage [19] and Spyglass Dicer are provided. By mapping solution values to color, one is able to see spatial variations of the solution variable much more rapidly. In addition one can animate sequences of frames, useful for simulating transient effects and for looking at dc sweeps. The translators from STRIDE data format to the format of these Visualization tools operate as stand-alone utilities.

VII. 3-D LATCH-UP ANALYSIS WITH STRIDE

To fully appreciate the benefits of 3-D device simulation, the relation of layout to latch-up conditions were investigated. Many studies in 2 D, such as [20] and [21], have shown that latch-up trigger current is lower than the expected value based on circuit models [22] due to debiasing under the P+ contact as shown in Fig. 5. The current flow in the n-well under the P+ induces a voltage drop. This voltage drop in turn forward biases the P+ /n-well junction, resulting in injected current into the

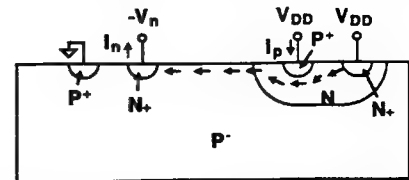


Fig. 5. Structure for 2-D latch-up analysis.

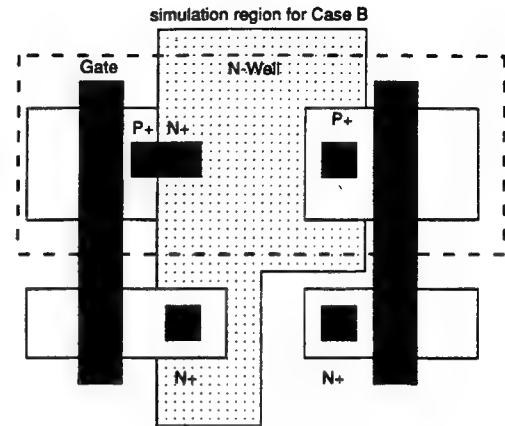


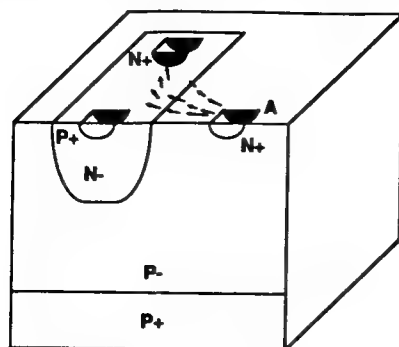
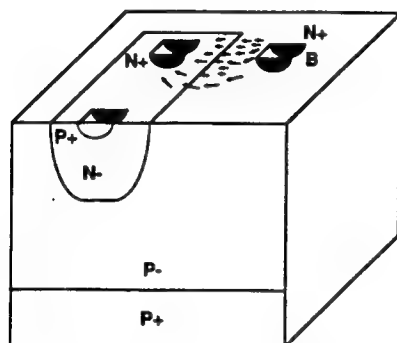
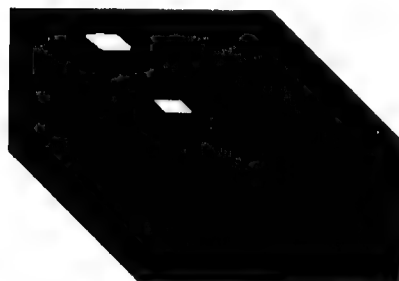
Fig. 6. CMOS layout.

substrate. The injected current into the substrate debiases the injector contact, resulting in sustained latch-up. In terms of the circuit model, lateral injection causes the vertical p-n-p to conduct, which in turn increases the biases on the lateral n-p-n.

The trigger current calculated from the 2-D structure can be judged as very conservative when compared to the case seen in an actual layout of an inverter as shown in Fig. 6, where all the injected current does not flow under the P+. Furthermore, the N+ S/D can face either a N-well contact (case B) as shown or a P+ S/D (Case A). For most standard cell layout, case B is more typical.

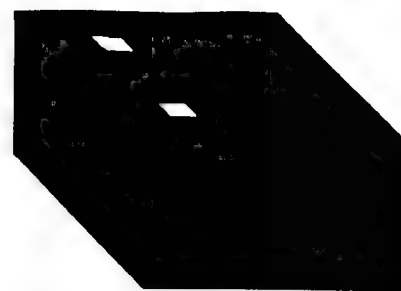
The simulation structure for 3 D contains about 10 000 nodes with dimensions of 10 μm in the x -direction, 10 μm in the y -direction, and 7 μm in the z -direction. A buried layer is used on the bottom of the structure. Four contacts are used—three on the surface, with two different injector contact positions as shown in Figs. 7 and 8, and one on the bottom acting as a substrate contact. Doping profiles are analytic functions for simplicity. The n-well and P+ contact inside the n-well are held at 2-V potential, while the substrate contact is held at 0 V. The voltage at the N+ injector is negatively biased until the current at the node dramatically increases and the carriers flood the device—indicating a latch-up condition. Using MSP-2C, the average time to simulate the trigger current is about 6 h on the 16 node hypercube, corresponding to the use of four bias points. The output data from STRIDE are post-processed into a format which is readable by the visualization tool Spyglass Dicer.

The positioning of the injector contact in the substrate is very important in determining the value of trigger current of this structure. Based on the debiasing mechanism

Fig. 7. 3-D latchup analysis structure: case *A*.Fig. 8. 3-D latchup analysis structure: case *B*.Fig. 9. 3-D plot of voltage drop at latchup onset: case *A*.

for the 2 D discussed above, it would seem plausible that the trigger current from case *A* would be lower than that from case *B* since more of the injected current would tend to flow under the P+ contact for case *A*. STRIDE simulations do indeed confirm this, as seen from Figs. 9 and 10. These 3-D pictures, taken from Spyglass Dicer, show a voltage drop (ψ minus its equilibrium value) as a function of the position prior to latch up (injected currents (Table I) are different). Differences in shading corresponds to gradients in voltage drop.

The main difference between the gray scale figures can be seen at the N+ well contact (top left corner of each of the images). It is not possible to examine the voltage drops at the P+ contact in the N-well since at the onset of latch-up the amount of debiasing is similar due to the exponential voltage dependence of current. The point to remember is that the device will latch up at a particular current flow in the well. For case *A*, the region around the

Fig. 10. 3-D plot of voltage drop at latchup onset: case *B*.TABLE I
LATCH-UP TRIGGER CURRENT

Trigger Current (mA)	
Case <i>A</i>	Case <i>B</i>
0.244	0.482

N+ contact in the direction away from the injector is dark, corresponding to an asymmetric current flow into the contact. This current is mostly traveling through the well in the direction of the P+ contact. In contrast, case *B* shows uniform shading around the N+ contact, showing that the injected current is being effectively collected by the both sides of the N+ contact. Thus case *B* should be collecting more total injected current at the onset of latchup. This finding is confirmed in Table I.

VIII. CONCLUSIONS

In this paper, we have reported progress in 3-D device simulation, focusing on computational aspects. By exclusively using iterative matrix solvers and insisting on low memory usage nonlinear iteration schemes, approximately 100 000 nodes can be solved on a user memory of about 100 M bytes. ICCG and ILUCGS types of iterative solvers are efficiently vectorized by using both the independent edge grouping scheme and the red-black ordering. Various MSP schemes are explored to improve the convergence of two carrier simulation using Gummel style nonlinear schemes. They not only offer significant improvement by themselves, but also provide the insight leading to the automatic selection of nonlinear schemes which offers the best CPU performance. Issues of the convergence criterion and global convergence scheme have also been successfully addressed with the insight provided by MSP schemes. A better user interface has been developed to facilitate program usage and tool integration. Aided with graphics capabilities made available through various graphics tool links, layout dependence of latch-up trigger current has been successfully analyzed.

ACKNOWLEDGMENT

The authors wish to thank Dr. Arthur Raefsky for discussions on vectorization schemes and visualization issues.

REFERENCES

- [1] T. Toyable, H. Masuda, Y. Aoki, H. Shukuri, and T. Hagiwara, "Three-dimensional device simulator CADDETH with highly convergent matrix solution algorithms," *IEEE Trans. Electron Devices*, vol. ED-32, pp. 2038-2043, Oct. 1985.
- [2] M. Thurner and S. Selberherr, "The extension of MINIMOS to a 3D simulation program," in *Proc. NASCODE V*, Dublin, 1987, pp. 327-332.
- [3] J. Burgler, P. Conti, G. Heiser, S. Paschedag, and W. Fichtner, "Three-dimensional simulation of complex semiconductor structures," in *Proc. Int. Symp. on VLSI Processes, Device and Systems*, Taiwan, 1989.
- [4] J.-H. Chern, J. T. Maeda, L. A. Arledge, Jr., and P. Yang, "SIERRA: A 3-D device simulator for reliability modeling," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 516-527, May 1989.
- [5] K.-C. Wu, R. F. Lucas, Z.-Y. Wang, and R. W. Dutton, "New approaches in a 3-D one-carrier device solver," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 528-537, May 1989.
- [6] J. W. Slotboom, "The p-n product in silicon," *Solid-State Electron.*, vol. 20, pp. 279-283, 1977.
- [7] D. M. Caughey and R. E. Thomas, "Carrier mobilities in silicon empirically related to doping and field," *Proc. IEEE*, vol. 55, pp. 2192-2193, 1967.
- [8] R. F. Lucas, private communication, Stanford Univ., Stanford, CA, 1988.
- [9] P. Conti, N. Hitchfeld, and W. Fichtner, "OMEGA—An octree-based mixed element grid allocator for adaptive 3D device simulation," in *Tech. Dig.—Workshop on Numerical Modeling of Processes and Devices for Integrated Circuits: NUPAD III*, Honolulu, Hawaii, June 1990.
- [10] D. Scharfetter and H. K. Gummel, "Large-signal analysis of a silicon Read diode oscillator," *IEEE Trans. Electron Devices*, vol. ED-16, pp. 64-77, 1969.
- [11] J. A. Meijerink and H. A. van der Vorst, "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," *Math. Computation*, vol. 31, no. 137, pp. 148-162, Jan. 1977.
- [12] C. den Heijer, "Preconditioned iterative methods for nonsymmetric linear systems," in *Proc. Int. Conf. on Simulation of Semiconductor Devices and Processes*, June 1984.
- [13] R. F. Lucas, K. Wu, and R. W. Dutton, "A parallel 3-D Poisson solver on a hypercube multiprocessor," in *Proc. IEEE Int. Conf. on Computer-Aided Design*, 1987, pp. 442-445.
- [14] R. F. Lucas, "Solving planar systems of equations on distributed-memory multiprocessors," Ph.D. dissertation, Stanford Univ., Dec. 1987.
- [15] F. Shakib, "Finite element analysis of the compressible Euler and Navier-Stokes equations," Ph.D. dissertation, Stanford Univ., Nov. 1988.
- [16] C. T. Kirk, Jr., "A theory of transistor cutoff frequency falloff at high current densities," *IRE Trans. Electron Devices*, pp. 164-174, Mar. 1962.
- [17] R. E. Bank and D. J. Rose, "Global approximate Newton methods," *Numer. Math.*, vol. 37, pp. 279-295, 1981.
- [18] W. Scheckler, A. S. Wong, R. H. Wang, G. Chin, J. R. Camagna, K. H. Toh, K. H. Tadros, R. A. Ferguson, A. R. Neureuther, and R. W. Dutton, "A utility-based integrated process simulation system," in *1990 Symp. on VLSI Technology—Dig. Tech. Papers*, June 1990, pp. 97-98.
- [19] "NCSA XImage for the X window system, version 1.0," National Center for Supercomputing Applications, Nov. 1989.
- [20] R. Menozzi, L. Selmi, E. Sangiorgi, G. Crisenza, T. Canioni, and B. Ricco, "Layout dependence of CMOS latchup," *IEEE Trans. Electron Devices*, vol. 35, pp. 1892-1901, Nov. 1988.
- [21] M. R. Pinto and R. W. Dutton, "Accurate trigger condition analysis for CMOS latchup," *IEEE Trans. Electron Device Letters*, vol. EDL-6, pp. 100-102, Feb. 1985.
- [22] D. B. Estreich, "The physics and modeling of latch-up and CMOS integrated circuits," Tech. Rep., G201-9, Stanford Electron. Labs, Stanford, CA, Nov. 1980.



Ke-Chih Wu (M'87) graduated from Fudan University, Shanghai, China in 1977, and received the M.S. and Ph.D. degrees in electrical engineering from Stanford University, CA, in 1981 and 1987, respectively.

From 1977 to 1978, he worked as a research assistant in the area of CCD technology with Technology and Physics Research Institute in Shanghai, China. Currently, he is a Research Associate in the Department of Electrical Engineering, Stanford University. His main research inter-

ests are in the area of multidimensional numerical analysis of semiconductor devices on parallel computers.



Goodwin R. Chin (S'84-M'85-S'86) received the B.S. degree in electrical engineering and computer science and materials science and engineering from the University of California at Berkeley in 1983, and the M.S. degree in materials science from Stanford University in 1984. He is currently working towards the Ph.D. degree in electrical engineering at Stanford University.

From 1984 to 1985 he worked at General Electric Corporate Research and Development, Schenectady, NY, in the areas of circuit design and advanced technology development. From 1985 to 1987 he worked at GE Intersil in the areas of analog and digital circuit design and advanced process development. From 1988 to 1989 he worked as a private consultant in the area of circuit design.

Mr. Chin is a member of Eta Kappa Nu and Tau Beta Pi.



Robert W. Dutton (S'67-M'70-SM'80-F'84) received the B.S., M.S. and Ph.D. degrees from the University of California, Berkeley, in 1966, 1967, and 1970, respectively.

He has held summer staff positions at Fairchild, Bell Telephone Laboratories, Hewlett-Packard and IBM Research in 1967, 1973, 1975, and 1977, respectively. He is currently a Professor of Electrical Engineering and Director of Research in the Center for Integrated Systems, Stanford University. He has published more than 200 articles in

technical journals. His research interests include integrated circuit process, device, and circuit technologies, especially the use of computer-aided design and parallel computational methods.

Dr. Dutton received the 1987 IEEE J. J. Ebers Award and the 1988 Geiggenheim Fellowship. From 1984 to 1986, he was the Editor of IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN.

New Approaches in a 3-D One-Carrier Device Solver

KE-CHIH WU, ROBERT F. LUCAS, ZE-YI WANG, AND ROBERT W. DUTTON, FELLOW, IEEE

Abstract—A 3-D one-carrier device solver has been developed on an Intel iPSC2™ hypercube multiprocessor which can handle over 130 K nodes. CPU time averages 20 min per bias point on a 50 K node MOSFET example. Slotboom variables are used in conjunction with the Scharfetter–Gummel current discretization scheme. A scaling scheme is proposed which produces n, p variables from the Slotboom variables. An improved damped-Newton scheme, which maintains the iteration numbers at below fifteen for high gate biases, is used in solving Poisson's equation. The performance of a previously proposed initial guess scheme is improved through the use of a novel update strategy during the Poisson solution stage after the initial guess step. This improvement allows stable calculation for voltage steps as high as five volts. A modified singular perturbation scheme (MSP) has been proposed whose implementation speeds up the convergence under high V_{gs} and V_{ds} bias conditions by a factor of three to six. A block matrix analysis of the MSP scheme yields insight into its performance.

I. INTRODUCTION

THE continuing reduction in VLSI device dimensions has made 3-D device simulation increasingly important. Unfortunately, the CPU time typically needed is so long that a supercomputer must often be used. Multiprocessors, with their high ratio of performance to cost, offer an attractive alternative to the use of supercomputers. To explore the potential for a cost effective 3-D device simulator, we have developed a prototype one carrier device solver on an Intel iPSC2™ hypercube. Another objective has been to explore ways to improve the robustness and efficiency in a device solver which uses exclusively iterative matrix solution methods. During this process, we have addressed several important issues related to device simulation in general, though with special emphasis in 3-D simulation.

Section II gives an overview of the device solver. Section III addresses issues in the use of Slotboom variables in the current continuity equations. Two scaling schemes are implemented to alleviate the overflow problem of the Slotboom variables. A scheme is proposed which produces the n, p variables as the scaled Slotboom variables. The Scharfetter–Gummel discretized current formulation is used. In Section IV, a simple and effective method for calculating the damping coefficient of the damped Newton scheme of Poisson's equation is presented. In Section V, the issue of initial guess is addressed. After an analysis

of the performance of a previously proposed initial guess scheme, a novel update strategy is proposed which, when used in the Poisson solution stage after the initial guess step, yields a consistent and dramatic reduction of the error in the current continuity equation. Section VI discusses methods for accelerating the Gummel iteration and presents a modified singular perturbation (MSP) scheme which results in a significant acceleration. In Section VII, the results for a MOSFET simulation are presented. Section VIII presents a block matrix analysis of the MSP scheme which give insight into the MSP scheme's performance. Finally, conclusions are drawn in Section IX.

II. OVERVIEW OF THE DEVICE SOLVER

In a one-carrier device solver, one of the two current continuity equations (assumed to be the electron current continuity equation for the discussion in this paper) is solved together with Poisson's equation. In normalized form, these two equations are given by

$$\nabla \cdot (\epsilon \nabla \psi) = n - p + N_A - N_D \quad (1)$$

$$\nabla \cdot J_n = 0 \quad (2)$$

where $n = \exp(\psi - \phi_n)$, $p = \exp(\phi_p - \psi)$, and $J_n = -q\mu_n n \nabla \phi_n$. The normalization constants used to obtain (1) and (2) are thermal voltage (kT/q) for electrostatic potential ψ and quasi-Fermi levels ϕ_n and ϕ_p ; intrinsic carrier concentration n_i for carrier and impurity concentrations and the intrinsic Debye length $\sqrt{(\epsilon kT)/(q^2 n_i)}$. Boltzman statistics are assumed as can be seen in the formula for n and p . Tabulated doping-dependent mobility values are used. At present, bandgap narrowing effects at high doping concentrations and field dependent mobility are neglected.

The development vehicle for the device solver is a multiple instruction stream, multiple data stream (MIMD), message passing, hypercube multiprocessor—the Intel iPSC2™. The advantages of the hypercube architecture are that it scales to massively parallel systems and that the diameter of the system (average communication delay between the processors) grows only logarithmically with the number of processors. An important feature of Intel hypercube is the amount of the memory it has. The system used in this work has 16 processors, each with 8 M Bytes of memory.

The simulation domain is approximated by a 3-D rectangular grid. Equations (1) and (2) are discretized by the finite difference method. To achieve parallel operation, the grid is decomposed into blocks of contiguous cubes

Manuscript received July 5, 1988; revised October 26, 1988. This work was supported by Intel Corporation, Texas Instruments, Inc., and by the U.S. Army Research Office under Contract DAAL03-87-K-0077. The review of this paper was arranged by Guest Editor M. Pinto.

The authors are with the Integrated Circuits Laboratory, Stanford University, Stanford, CA 94305.

IEEE Log Number 8816893.

that are allocated to each of the processors. Different material properties are specified by assigning different material parameters to the volumes (dielectric constant, intrinsic carrier concentration, etc.). The vertices that are shared by multiple processors are labeled separators. Further details of the grid partition are referred to in [1]. Nonplanar structures are handled by a three-dimensional extension of the scheme used in the 2-D device simulation program GEMINI [2]. The volume associated with each edge that originates at a vertex is a pyramid that can be further decomposed into two tetrahedra. By specifying different material properties for each of the tetrahedra, a cube can be subdivided into regions of different material. The Jacobi matrices from the finite difference equations are assembled by summing the contributions from each volume. Since each processor is allocated a unique set of cubes, the sparse systems of equations can be assembled without communication. Coefficients of the equations corresponding to the separator vertices are stored locally on the processors that generate them.

The discretization of (1) and (2) yields a nonlinear system of algebraic equations which are solved by one of the two nonlinear iteration methods implemented in the solver. In each nonlinear iteration, Gummel's method first solves the discretized Poisson's equations ($F_\psi(\psi, \Phi_n) = 0$) for the update vector $\delta\psi$ holding Φ_n constant.¹ This is achieved by repeatedly solving

$$A_{\psi\psi}\delta\psi = -F_\psi \quad (3)$$

given the current estimate of ψ and Φ_n . In (3), $A_{\psi\psi} \equiv (\partial F_\psi(\psi, \Phi_n))/\partial\psi$ is called the main matrix of Poisson's equation. The discretized current continuity equations ($F_{\Phi_n}(\psi, \Phi_n) = 0$) are then solved for the update vector $\delta\Phi_n$ holding ψ constant

$$A_{\Phi_n\Phi_n}\delta\Phi_n = -F_{\Phi_n} \quad (4)$$

where $A_{\Phi_n\Phi_n} \equiv (\partial F_{\Phi_n}(\psi, \Phi_n))/\partial\Phi_n$ is called the main matrix of electron current continuity equation. This process, also called Gummel iteration, repeats until convergence. In the block-iterative Newton's method [3], [4], an update to the solution is generated by repeatedly solving a system of linear equations derived from the current estimates of both the solution vectors (ψ, Φ_n).¹ Using block matrix formulation, these equations can be written as

$$Ax = -F \equiv \begin{bmatrix} A_{\psi\psi} & D_{\psi\Phi_n} \\ A_{\Phi_n\psi} & A_{\Phi_n\Phi_n} \end{bmatrix} \begin{pmatrix} \delta\psi \\ \delta\Phi_n \end{pmatrix} = - \begin{pmatrix} F_\psi \\ F_{\Phi_n} \end{pmatrix} \quad (5)$$

where $D_{\psi\Phi_n} \equiv (\partial F_\psi(\psi, \Phi_n))/\partial\Phi_n$ and $A_{\Phi_n\psi} \equiv (\partial F_{\Phi_n}(\psi, \Phi_n))/\partial\psi$ are called the coupling matrices of Poisson's and the electron current continuity equations, respectively. $D_{\psi\Phi_n}$ denotes that it is a diagonal. The convergence criteria are the maximum magnitude of ψ updates, terminal current conservation, relative change in the magnitude of terminal currents and of terminal charges. The terminal

current criteria are found to be important in the low current regime where ψ converges faster than the terminal currents.

The matrix solutions are the most CPU time intensive steps in the device solver. To solve symmetric matrices, the incomplete Cholesky conjugate gradient (ICCG) algorithm is used. The concurrent implementation of ICCG algorithm is referred to in [1]. To solve asymmetric matrices, two algorithms are currently used. One is the conjugate gradient squared (CGS) algorithm [5]. The other is the generalized minimal residual (GMRES) algorithm [6] with up to eighteen back vectors. The number of back vectors is limited by the memory constraints. For both algorithms, the ILU preconditioner is used. The concurrent implementation of these two algorithms and the ILU preconditioner is a simple extension of that of ICCG algorithm. In our experience, the ILU-CGS algorithm converges faster but the residual tends to oscillate. So far, the final convergence has always been achieved. On the other hand, with the ILU-GMRES algorithm, the residual decreases monotonically but the overall convergent rate is slower than that of the ILU-CGS algorithm.

The maximum number of grid points that can be handled by the solver on our 16-node system is over 130 000, which translates to a cubic grid of 51 points in each dimension. CPU time per bias point is about 20 min for a 50 K node MOSFET example. This is averaged from three I_{ds} versus V_{ds} curves with $V_{gs} = 1, 3$, and 5 V. In each case, V_{ds} increases from 1 to 5 V in 1 V steps.

III. USE OF SLOTBOOM VARIABLES

Slotboom variables [7] invoke the exponential of the quasi-Fermi levels as the independent variables:

$$\Phi_n = \exp(-\phi_n); \quad \Phi_p = \exp(\phi_p).$$

The advantages of using the Slotboom variables are that [8] the current continuity equation becomes linear and that its matrix becomes both positive definite and symmetric. Iterative solution techniques needed for a system of nonlinear equations are thereby avoided. Furthermore, symmetric matrices take less time to solve than asymmetric ones. This advantage becomes more significant for iterative matrix solution methods. For low-bias applications where Gummel iteration converges rapidly, using the Slotboom variables offers higher computational efficiency than other alternatives. The disadvantage of using the Slotboom variables is that these variables overflow at large voltage bias. Using IEEE standard double precision capabilities, the maximum number which can be represented is about 10^{300} . However, even this huge number can only accommodate a bias of about 18 V at room temperature. At low temperatures, the maximum voltage is further reduced, to less than 5 V at 77 K.

The variable overflow results from the separation between ψ and the quasi-Fermi levels. We use two scaling schemes which preserve the symmetry of the matrix to increase the bias range. The first scheme is to symmetrically scale the matrix for the current continuity equation

¹ Φ_n indicates the use of Slotboom variable in the continuity equation which will be discussed in Section III. The discussion of this section is independent of what variable is used in the continuity equation.

which involves scaling both the variables and the right hand side of the equations. For the case of electrons, we have

$$A_{\Phi_n \Phi_n} \delta \Phi_n = -F_{\Phi_n} \rightarrow (\exp(-\psi/2) A_{\Phi_n \Phi_n} \exp(-\psi/2)) \cdot \delta y_n = -\exp(-\psi/2) F_{\Phi_n}$$

where δy_n is the update vector for $y_n (\equiv \exp(\psi/2) \Phi_n)$, the variable vector after scaling. The scaling factor $\exp(-\psi/2)$ is a diagonal matrix whose i th diagonal is $\exp(-\psi_i/2)$. For the case of holes, $\exp(\psi/2)$ should be used instead. The second scheme further scales the variables produced by the first scheme. The objective is to center them around a value of one. If we denote these scaled variables as w , then for the case of electrons,

$$w_n = y_n \exp((\phi_{n,\max} + \phi_{n,\min})/4) = \exp((\psi - \phi_n)/2) \cdot \exp(-\phi_n/2 + (\phi_{n,\max} + \phi_{n,\min})/4). \quad (6)$$

A similar formula applies for holes. Each scheme approximately doubles the bias range. For silicon, the maximum bias at room temperature is about 70 V and that at 77 K is 16 V. These biases are adequate for most applications.

For high voltage applications, the use of n, p variables seems to be inevitable. A straightforward use of n, p variables, however, poses a problem for the accelerating schemes of Gummel's method such as the MSP scheme proposed in this paper. It is well known that when using the n, p variables with Gummel's method, the matrix of Poisson's equation still needs to be constructed using either the Slotboom or the quasi-Fermi level variables, a switch from the n, p variables used in the current equation [9]. Since schemes like the MSP scheme make use of the coupling matrices appeared in (5), the question of what variables one should use to construct these matrices arises. We have worked out a scheme to resolve this issue in which the n, p variables are obtained by properly scaling the matrix obtained with the Slotboom variables. We get

$$\begin{aligned} & \begin{bmatrix} A_{\psi\psi} & D_{\psi\Phi_n} \\ A_{\Phi_n\psi} & A_{\Phi_n\Phi_n} \end{bmatrix} \begin{pmatrix} \delta\psi \\ \delta\Phi_n \end{pmatrix} \\ & \equiv \begin{bmatrix} A_{\psi\psi} & D_{\psi\Phi_n} \\ A_{\Phi_n\psi} & A_{\Phi_n\Phi_n} \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & \exp(-\psi) \end{bmatrix} \begin{pmatrix} \delta\psi \\ \delta n \end{pmatrix} \\ & = - \begin{pmatrix} F_{\psi} \\ F_{\Phi_n} \end{pmatrix}. \end{aligned} \quad (7)$$

The scaling factor for holes is $\exp(\psi)$. It is clear from (7) that both $A_{\psi\psi}$ and $A_{\Phi_n\psi}$ remain unchanged by the scaling. This explains the above-mentioned, seemingly strange situation of having to switch variables between Poisson's and the current continuity equations. The answer is that the n, p variables used with Gummel's method are actually these scaled Slotboom variables. More significantly, the Gummel iteration acceleration schemes such as the MSP scheme can now be applied to these " n, p " variables without difficulty.

Due to its stability and improved accuracy in the presence of rapidly varying carrier concentrations [10], we have chosen to incorporate the Scharfetter-Gummel (S-G) discretized current formulation [11] with the Slotboom variables. The discretized current formula without the S-G scheme is

$$\begin{aligned} J_{n,i+1/2} & = q\mu_{n,i+1/2} \exp((\psi_i + \psi_{i+1})/2) (\Phi_{n,i+1} - \Phi_{n,i}). \end{aligned} \quad (8)$$

As shown by Bank *et al.* [3], using the S-G formula amounts to replacing $\exp((\psi_i + \psi_{i+1})/2)$ with $(\psi_{i+1} - \psi_i)/(\exp(-\psi_i) - \exp(-\psi_{i+1}))$ and results in

$$\begin{aligned} J_{n,i+1/2}^{S-G} & = q\mu_{n,i+1/2} \frac{\psi_{i+1} - \psi_i}{\exp(-\psi_i) - \exp(-\psi_{i+1})} \\ & \cdot (\Phi_{n,i+1} - \Phi_{n,i}) \equiv B((\psi_{i+1} - \psi_i)/2) J_{n,i+1/2} \end{aligned} \quad (9)$$

where $B(x) \equiv x/\sinh(x)$ is equation (56a) in [3]. $B(x)$ can be thought of as a correction coefficient to (8) and can be called symmetric Bernoulli function, since it is similar to the Bernoulli function [3] but is an even function of x . $B(x)$ never becomes greater than one, which means that the S-G formula (9) always reduces the current density estimated by (8).

It is noteworthy that although the S-G formula has originally been derived with the assumption of constant current density along the current path, an alternative derivation without such an assumption is possible. In addition, the truncation error associated with $J_{n,i+1/2}^{S-G}$ is also of the order h^2 , just as it is in the case of the other methods. The details are described in the Appendix.

IV. AN IMPROVED DAMPED-NEWTON SCHEME FOR POISSON'S EQUATION

Traditionally, the update vector obtained from solving the linearized equations has been applied in full, regardless of its effect on the residual vector of the nonlinear equations. Although convergence can usually be achieved in a reasonable number of iterations, there are occasions when an excessive number is required. By monitoring both the maximum ψ update in magnitude $\delta\psi_{\max}$ and the infinity norm of the residual vector $e_{\psi,\max}$, we found that slow convergence is always accompanied by a large jump of $e_{\psi,\max}$ (often of many orders of magnitude) at an early Newton iteration. This is followed by a slow decay of $e_{\psi,\max}$ with a linear decay rate of e^{-1} before the residual enters the final quadratic convergence phase. $\delta\psi_{\max}$ is, for the most part, very close to unity during this slow decline phase. Clearly, elimination of the growth of $e_{\psi,\max}$ by damping the update vector would greatly shorten this slow decay phase. However, the classical scheme for finding the damping coefficient [12] has proved to be ineffective

in that overdamping cannot be easily avoided without trial and error.

To find an effective scheme for estimating the damping coefficient which allows the maximum update without overshooting the residual vector, one should look more closely at the source of the problem. The linearized Poisson's equation at node j can be written as

$$F_{\psi,j} = \sum_k c_{jk}(\delta\psi_j - \delta\psi_k) + (n_j + p_j)\delta\psi_j. \quad (10)$$

The first term is the contribution of the Laplace operator ∇^2 and it is a linear function of $\delta\psi$. The second term is the contribution of the mobile carriers and is nonlinear due to the exponential dependence of carrier concentrations on ψ . Since only the nonlinear part of Poisson's equation can cause such a problem, it becomes apparent that a large $e_{\psi,\max}$ overshoot must be accompanied by a major overestimation of the carrier concentrations. In physical terms, the overshoot is the result of the failure of the linearized Poisson's equations to predict the inversion charge induced by an increase in gate bias. Since a change in ψ does not change the np product, ($np = \exp(\phi_p - \phi_n)$), e_{ψ} at the nodes of large overshoot is dominated by either electron or hole concentrations. In other words, $e_{\psi,i}$ can be assumed to be proportional to $\exp(|\psi_i|)$ at these nodes. With this simplification, the damping coefficient α which prevents the growth in $e_{\psi,\max}$ can be easily estimated. Let $e_{\psi,\max}^k$ denote $e_{\psi,\max}$ at the k th Newton iteration and $\delta\psi_{im}^* \equiv |\delta\psi_{im}|$ where node i_m is the maximum residual node. When $e_{\psi,\max}^{k+1} > e_{\psi,\max}^k$, the amount of reduction needed in $\delta\psi_{im}^*$ to eliminate this overshoot can be estimated as $1 + \ln(e_{\psi,\max}^{k+1}/e_{\psi,\max}^k)$. The damping coefficient α is thus

$$\alpha = 1 - \frac{1 + \ln(e_{\psi,\max}^{k+1}/e_{\psi,\max}^k)}{\delta\psi_{im}^*}. \quad (11)$$

The unity term in the numerator is added to compensate for the inaccuracy in the error model. After α is obtained from (11), it is used to damp the entire update vector. This process is repeated if a feasible α is not found in the first try ($e_{\psi,\max}^{k+1}$ is still greater than $e_{\psi,\max}^k$). This scheme has proved very effective in speeding up the convergence of Poisson's equation. In most cases, only a single try is needed to find a feasible α . Fig. 1(a) shows the evolution of $e_{\psi,\max}$ on a log scale for a MOSFET example with three values of V_{gs} , while Fig. 1(b) shows the damping coefficients used in these iterations. Since the threshold voltage of this device is about 0.75 V, the value of V_{gs} varies from 1 V, slightly above V_T , to 5 V, well into the strong inversion. All the damping coefficients, which are less than one, are calculated with a single try. As Fig. 1(a) indicates, the number of iterations increases only from 9 to 11 as V_{gs} changes from 1 to 5 V. This result becomes more significant due to the fact that without damping, the $e_{\psi,\max}$ overshoot at $V_{gs} = 1$ V would be only a factor of 10^2 while that at $V_{gs} = 5$ V it would be 10^{28} ! In addition, Fig. 1(b) shows that the change in the feasible α with the iteration

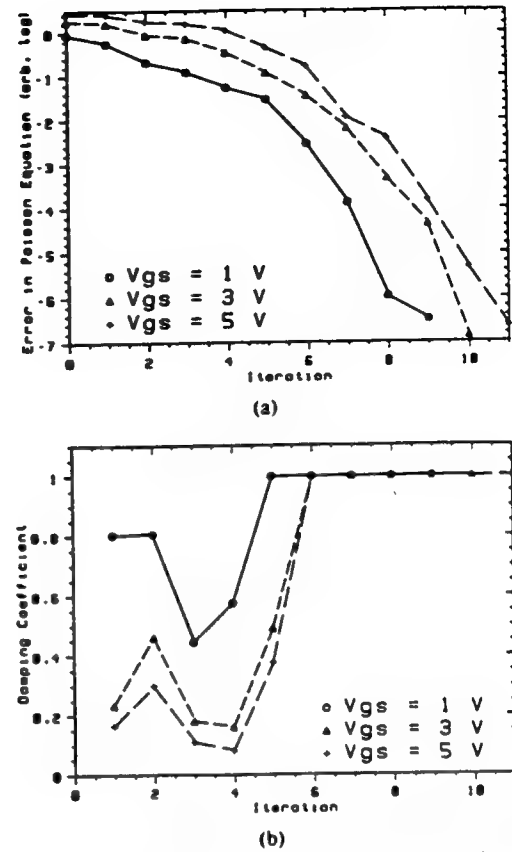


Fig. 1. (a) Poisson equation error versus iteration. (b) Damping coefficient versus iteration.

is not smooth. Therefore, schemes based on the assumption that the change in α is smooth would not do as well. As the result of the success of this scheme, we have not attempted more involved schemes, such as selecting different damping coefficients for each node.

V. INITIAL GUESS SCHEME

Traditionally, there is no initial guess² provided by a device simulator at the first bias step and only guesses of limited utility³ are available at the subsequent bias steps. Hence, the solution process usually begins right after the boundary nodes have been updated to the present bias conditions. As pointed out by Rafferty [8], extremely large updates can be generated, updates so large that even conventional damping schemes fails to make progress towards the solution.

In recent developments [15]–[17], various initial guess steps have been proposed which precede the iterative solution process. In a scheme which is applicable to Gummel iteration [15], [16], the total current continuity equation is solved for a change in the potential distribution $\delta\psi$, with carrier concentrations held at the values obtained

²Here, an initial guess means a solution other than the solution at equilibrium or that of the previous bias with boundary nodes are updated with present bias. Therefore, the *previous* initial guess in PISCES program [13] means no initial guess in this context.

³Among these guesses, the linear extrapolation from a pair of previous solutions has proven to be the most effective in general [8]. However, its advantage over the use of previous solution is not universal [14].

from the previous solution:

$$\nabla \cdot ((\mu_n n + \mu_p p) \nabla \delta \psi) = 0. \quad (12)$$

The electrostatic potential ψ and the quasi-Fermi levels, ϕ_n and ϕ_p , are then updated from their values from the previous solution according to $\delta \psi$. Then, Gummel iteration is started by first solving Poisson's equation, which is ignored in (12).⁴

The constant carrier concentration assumption is acceptable in the charge neutral region as long as high level injection is not present. This scheme is, therefore, quite successful in producing adequate initial guesses for the charge neutral region. In fact, the resistor problem, which is notoriously difficult to solve with Gummel's method [9], can be solved by this scheme alone. However, this scheme fails to predict the change in the space-charge region resulted from the bias. When a large bias step is applied, this may cause problems during the Poisson solution stage after the initial guess step. Fig. 2 is a schematic of what can happen around the drain-substrate junction of an n-channel MOSFET device when a large bias step is applied. The lower solid curve sketches the ψ distribution at equilibrium while ϕ_n is zero everywhere. The dashed curve sketches ϕ_n obtained with the initial guess. Because of the initially narrow space-charge region, ϕ_n drops very rapidly across the junction. On the other hand, the ψ distribution in the final solution, which is sketched as the upper solid curve, drops less rapidly across the junction because of the widened space-charge region. What is the implication of the steep drop of ϕ_n across the junction? Since $n = \exp(\psi - \phi_n)$, the amount of separation between "new ψ " and ϕ_n near the origin of x axis indicates the doping concentration in the heavily doped drain region. Notice that somewhere in the space-charge region, "new ψ " exceeds ϕ_n by much more than this amount. This means that if ψ were to assume its final solution value in the first Poisson solution stage following the initial guess, the electron concentration in the junction region could be much higher than the doping concentration in the drain region. Instead, ψ is forced to follow ϕ_n to avoid such an overshoot. This has an important impact on the convergence process. As the difficulties with the resistor problems suggest, when large errors occur in the charge neutral region where ψ follows the quasi-Fermi level of the majority carrier closely, Gummel iteration becomes very slow. Therefore, an artificially created high concentration region will significantly slow down the solution process.

Clearly, large bias performance of the initial guess scheme can be improved if this artificial high-concentration region can be eliminated. As the above discussion indicates, the problem arises because the initial guess scheme gives an erroneous distribution of the quasi-Fermi levels. Therefore, the logical solution is to somehow modify the quasi-Fermi levels when necessary. This calls

⁴If carrier concentration is constant everywhere in the simulation domain, then Poisson's equation is also implied by (12). Otherwise, the solution from (12) will not satisfy Poisson's equation.

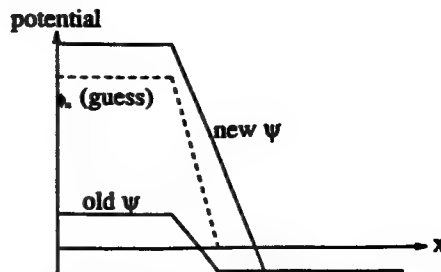


Fig. 2. Schematic illustration of potential distribution around drain-substrate junction with a large bias step.

for a special update strategy during the Poisson solution stage: if carrier concentration overshoots when ψ receives its update $\delta \psi$, the appropriate quasi-Fermi level is modified to prevent this overshoot. More specifically, the increase of majority carrier concentration at a node j is limited to be proportional to $\delta \psi_j$ by adjusting $\phi_{n,j}$ or $\phi_{p,j}$ when necessary. For example, at the k th iteration, if $\delta \psi_j^k > 1$ (which means n_j will significantly increase) and if $n_j^{k+1} > p_j^{k+1}$ (which means electron is the majority carrier at node j), then $\phi_{n,j}^k$ will be increased such that $n_j^{k+1} = (1 + \delta \psi_j^k) n_j^k$. A similar procedure applies for adjusting $\phi_{p,j}^k$ when $\delta \psi_j^k < -1$. For the example mentioned in Fig. 2, ϕ_n in the space-charge region will be raised to avoid the electron concentration overshoot. One justification for this linear increase is as follows: as can be seen from (10), the linearized Poisson's equation expects a carrier concentration change at node j which is proportional to $\delta \psi_j$.⁵ This reasoning kept us from choosing an alternative scheme of maintaining the starting carrier concentration. It should be emphasized that this scheme only prevents the overshoot of carrier concentrations during the Poisson solution stages, and is no substitute for actually solving the current continuity equation.

In performing MOSFET simulations where V_{gs} is changing, a Poisson solution is first performed with the gate voltage alone updated. The initial guess scheme discussed above is then applied. With this "double initial guess" scheme, the nonlinear effects caused by the changes in V_{gs} and V_{ds} are dealt with separately. With this implementation, a dramatic reduction (by several orders of magnitude) in the error of the current equation is consistently observed after the first Poisson solution, and convergence is achieved for an initial bias step of five volts on both gate and drain for an n-channel MOSFET example. More details are provided in Section VII where convergence results are presented.

VI. ACCELERATION OF GUMMEL ITERATION

Among the possible solution schemes for the coupled Poisson and current continuity equations, Gummel's

⁵In fact, this is exactly why the slow decay process, mentioned in the last section, is always associated with a $\delta \psi_{max}$ of one. As mentioned in the last section, carrier concentration dominates the residual term at the overshooting nodes. If too many electrons are at such a node i_m , then, with a $\delta \psi_{im}$ approaching -1 , the linearized Poisson's equation expects to eliminate almost all the electrons at that node. Instead, this only causes $n_{im}^{k+1} = e^{-1} n_{im}^k$. This process goes on until n_{im} no longer dominates $e_{\psi,im}$.

method offers the advantage of requiring the least amount of memory and the best computational efficiency at low level injection [8]. However, since the coupling between the Poisson and current continuity equations is ignored, the method's performance deteriorates quickly at high level injection. Therefore, there have been efforts to accelerate the Gummel iteration in the high level injection region.

Two schemes have recently been proposed to accelerate the Gummel iteration. Blakey [18] proposed the use of a Concus-Golub transformation with Slotboom variables in the current continuity equation and reported a significant extension of the region over which the Gummel iteration is computationally advantageous. A singular perturbation (SP) scheme was proposed by Ringhofer [19], who also reported a significant improvement in the convergence properties of the Gummel iteration.

It is interesting to note that the Concus-Golub transformation is equivalent to the symmetric scaling of the matrix of Slotboom variables mentioned in Section III. However, our experience with this transformation in the MOSFET simulation has been disappointing. The convergence of the Gummel iteration is still painfully slow in the high gate and drain bias region. The discrepancy in performance between our results and those previously reported may be attributed to the difference between a MOSFET and a bipolar device (either a diode or a transistor) at high level injection. In a bipolar device, high level injection causes the minority carrier concentration to be roughly equal to the majority carrier concentration in the active regions of the device where charge neutrality still dominates. In a MOSFET, however, high level injection creates an inversion layer where the charge imbalance is exacerbated by an increase in mobile carriers. This makes ψ more susceptible to any change in mobile carrier concentration and further reduces the convergence rate of the Gummel iteration.

Having had little success with the scaled Slotboom variable approach, we investigated the SP scheme. To illustrate the idea behind the SP scheme, we take a look at the complete linearized electron continuity equation from (5)

$$A_{\Phi_n \Phi_n} \delta \Phi_n + A_{\Phi_n \psi} \delta \psi = -F_{\Phi_n} \quad (13)$$

Comparing (4) and (13), we can see that Gummel's method ignores the coupling terms involving $A_{\Phi_n \psi}$. Therefore, convergence becomes slow when these coupling terms become significant. To improve the situation, the SP scheme partially takes these coupling terms into account by including an approximate $\delta \psi$ ($\delta \tilde{\psi}$) in (13). We get

$$A_{\Phi_n \Phi_n} \delta \Phi_n + A_{\Phi_n \psi} \delta \tilde{\psi} = -F_{\Phi_n} \quad (14)$$

where $\delta \tilde{\psi}$ is determined by a singular perturbation approximation to Poisson's equation at each node j which is given by [19]

$$(n_j + p_j) \delta \tilde{\psi}_j + n_j \frac{\delta \Phi_{n,j}}{\Phi_{n,j}} = -F_{\psi,j} \quad (15)$$

such that

$$\delta \tilde{\psi}_j = -\frac{n_j}{n_j + p_j} \frac{\delta \Phi_{n,j}}{\Phi_{n,j}} - \frac{F_{\psi,j}}{(n_j + p_j)} \quad (16)$$

Substituting (16) into (14), we obtain a modified matrix and right-hand side for the electron current continuity equation. Unlike the traditional Gummel iteration, the current equation matrix now becomes asymmetric and some extra work is needed to construct the modified matrix. The rest of the nonlinear iteration procedure is the same as in the traditional Gummel iteration. Obviously, the success of the scheme depends on how closely $\delta \tilde{\psi}$ approximates $\delta \psi$. To explain why a good approximation is expected, we compare the matrix form of (15)

$$D_{\psi\psi}^{\text{SP}} \delta \tilde{\psi} + D_{\psi\Phi_n} \delta \Phi_n = -F_{\psi} \quad (17)$$

with the complete linearized Poisson's equation from (5)

$$A_{\psi\psi} \delta \psi + D_{\psi\Phi_n} \delta \Phi_n = -F_{\psi} \quad (18)$$

The only difference between (17) and (18) is that in (17), $A_{\psi\psi}$ is replaced by $D_{\psi\psi}^{\text{SP}}$. Being a diagonal matrix with $D_{\psi\psi,j}^{\text{SP}} = n_j + p_j$, $D_{\psi\psi}^{\text{SP}}$ is just $A_{\psi\psi}$ minus the contribution of Laplace operator to $A_{\psi\psi}$. The argument for the SP scheme is thus: since the mobile carrier terms almost always dominate the diagonal of $A_{\psi\psi}$, which makes $A_{\psi\psi}$ approximately a diagonal matrix, one can use $D_{\psi\psi}^{\text{SP}}$ to approximate $A_{\psi\psi}$.

Our experience with this original version of the SP scheme in simulating MOSFETs is that it is rather unstable. As (16), the key formula of the SP scheme, indicates, whenever n_j is sufficiently larger than p_j , $\delta \tilde{\psi}_j$ follows closely the relative change in $\Phi_{n,j}$. This means a close coupling between Poisson's equation and the electron current continuity equation at node j , and hence, a strong constraint for n_j to remain unchanged. Although this is certainly true for a node in an n-type charge neutral region, it is not true for a node in the space-charge region where Poisson's equation is controlled by impurity charge rather than mobile carrier charge. Therefore, in the space-charge region, the coupling between the two equations is overestimated by the SP scheme. It is well known that when a problem is overly constrained, there may not be a feasible solution. In the matrix terms, $D_{\psi\psi,j}^{\text{SP}}$ may be negligible in comparison with the matrix elements in j th row of $A_{\psi\psi}$ if node j is in a space-charge region. Therefore, $\delta \tilde{\psi}_j$ is a very poor approximation of $\delta \psi_j$. A block matrix analysis of the SP scheme will be presented in Section VIII which gives mathematical reasons for the instability.

Based on above understanding, we propose a modified singular perturbation (MSP) scheme. In this scheme, the equivalent formula for determining $\delta \tilde{\psi}_j$ becomes

$$\left(n_j + p_j + V_j^{-1} \sum_k \frac{\epsilon_{jk} A_{jk}}{d_{jk}} \right) \delta \tilde{\psi}_j + n_j \frac{\delta \Phi_{n,j}}{\Phi_{n,j}} = -F_{\psi,j} \quad (19)$$

The added terms are the contributions of the Laplace operator to the diagonal of $A_{\psi\psi}$. The equivalent of (17) is

thus

$$D_{\psi\psi}\delta\bar{\psi} + D_{\psi\Phi_n}\delta\Phi_n = -F_{\psi} \quad (20)$$

where $D_{\psi\psi}$ is the diagonal of $A_{\psi\psi}$. For a node in a charge-neutral n-type region, the added terms are usually negligible compared to the mobile carrier terms, and the close coupling between the two equations is preserved. For a node j in the space-charge regions, however, the additional terms will dominate the coefficient for $\delta\bar{\psi}_j$ and the coupling between the equations is significantly reduced.

With this simple modification, we observed a significant improvement in convergence properties over traditional Gummel iterations. One practical issue for any solution method of nonlinear equations is the appropriate damping scheme. For Gummel's iteration, maintaining the nonnegativeness of carrier concentrations is sufficient. For the MSP scheme, it is also necessary to limit the relative change of $\delta\Phi_n$ at those nodes where Poisson's equation is closely coupled with the continuity equation. Currently, this limit is set at 50. Given node j and $\delta\Phi_{n,j}$, $\delta\bar{\psi}_j$ calculated from (19) may not exceed 50. Otherwise, $\delta\Phi_{n,j}$ is damped in such a way to make $\delta\bar{\psi}_j = 50$.

VII. CONVERGENCE RESULTS

To give a practical perspective on the proposed schemes, this section presents the relevant convergence results from the simulation of an n-channel MOSFET example.

The simulated MOSFET, shown in Figure 3, is a simplified one-micron minimum geometry device. Analytical (Gaussian or complementary error function) doping profiles are used to supply the impurity distribution. The gate oxide thickness is 250 Å and the substrate acceptor concentration is $2 \times 10^{16} \text{ cm}^{-3}$. The channel implant is a Gaussian profile with a peak concentration of 10^{17} cm^{-3} and a characteristic length of $0.2 \mu\text{m}$. Peak concentrations in the source and drain are 10^{20} cm^{-3} and the overlap between gate and source and drain regions is about $0.2 \mu\text{m}$. The total number of grid points used in the simulation is 5520 with 23 in the source-drain direction, 10 in the width direction, and 24 in the direction normal to the surface. The main reason for using a relatively small number of nodes is to test the stability of the solution scheme under adverse gridding conditions. We have observed a drop in the electron quasi-Fermi level of more than four volts between two neighboring nodes at $V_{ds} = 5 \text{ V}$.

Table I shows a comparison of the Gummel and MSP methods for several initial biases. V_{gs} is set to 5 V to simulate the high level injection condition, and three values of V_{ds} are used to observe the effect of the increasing bias step. Since with the MSP scheme, the matrices take longer to solve, both the iteration number and the ratios of the CPU times needed for these iterations are shown. Moreover, the MSP method is started after several traditional Gummel iterations, hence the number of iterations shown are counted after the MSP regime has started. It is immediately clear that the MSP scheme always takes a much smaller number of iterations to converge than does the

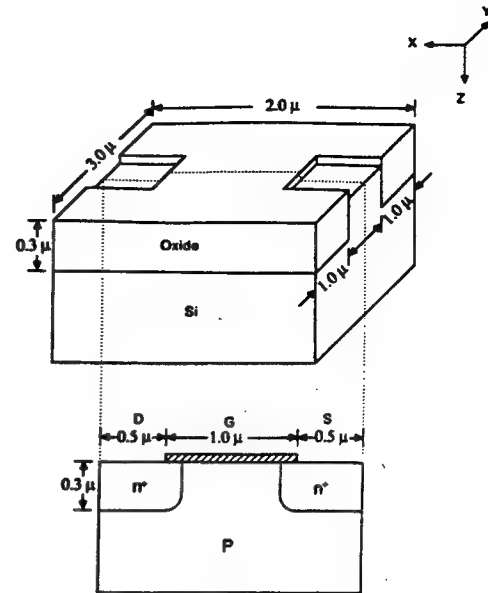


Fig. 3. Schematic representation of the simulated device.

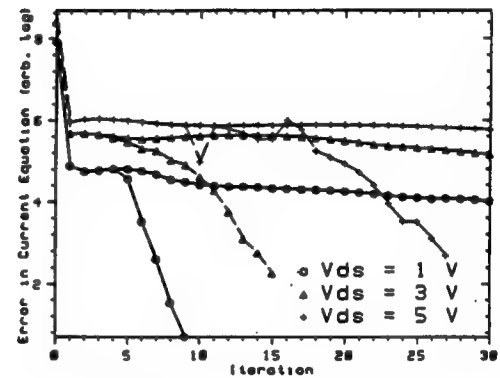


Fig. 4. Current equation error versus iteration ($V_{gs} = 5 \text{ V}$).

TABLE I
EFFECT OF MSP MODIFICATION IN A MOSFET EXAMPLE ($V_{gs} = 5 \text{ V}$)

Vds	# of iterations		CPU speedup	
	Gummel	MSP	Par. Soln.	TOTAL
1V	74	6	6.4	3.9
3V	136	13	4.1	3.4
5V	166	21	3.0	2.6

traditional Gummel iteration. Even after taking the longer matrix solution time into account, the CPU time ratio is still an impressive 6.4 at $V_{ds} = 1 \text{ V}$. The effectiveness of the MSP scheme deteriorates, however, as V_{ds} increases. The reason for the deterioration will be discussed in Section VIII. Total CPU time is the time needed for the program to run from beginning to the finish and includes all the overhead of the program. The ratios of total CPU time shown in the last column can be improved somewhat by calculating several bias steps at once.

Fig. 4 shows on a log scale how the error in the current equation evolves during the iterations (including the ini-

tial guess step). The effect of the initial guess step can be seen from the sharp drop of the error at the first iteration. The onset of the MSP regime can be seen at the points where the curves corresponding to Gummel iteration separate from that of the MSP iteration. For the iterations in which the MSP iteration has converged, the additional reduction of error for Gummel iteration is still very small.

VIII. BLOCK MATRIX ANALYSIS OF THE MSP SCHEME

Although the reasons for proposing the MSP scheme have been discussed in Section VI, a formal analysis of the scheme is lacking there. In proposing the alternate block factorization (ABF) scheme, Bank and Smith [20] have elegantly formulated their scheme using a block matrix formulation. Using this formulation, the ABF scheme can be represented by applying a transformation matrix (T_{ABF}) to the right side of the original matrix, with each block of T_{ABF} the transformation matrix being diagonal. Although block matrix analysis is usually applied to block iterative schemes such as ABF, Gummel iteration can be also viewed as a "poor man's" block iteration in which the influence of the coupling matrices is accounted for implicitly by reevaluating the right-hand side every time. It is therefore beneficial to have a block matrix formulation of the MSP scheme both for comparing the two schemes and for further understanding its properties.

For the equations to stay the same, a transformation matrix applied on the right of the matrix will also have to transform the variables. On the other hand, a transformation matrix applied on the left of the matrix will also have to modify the right-hand side to yield the same solution. The ABF scheme is an example of applying a transformation matrix on the right of the matrix.

The MSP scheme, however, must have its transformation matrix applied on the left of the matrix since the right-hand side is also modified. That is,

$$T_{MSP}Ax = -T_{MSP}F \quad (21)$$

where A and F are defined in (5). From this general format, we have formulated the following transformation matrix:

$$T_{MSP} = \begin{bmatrix} I & 0 \\ -A_{\phi_n\psi}D_{\psi\psi}^{-1} & I \end{bmatrix} \quad (22)$$

The matrices on the main diagonal have been chosen to be identity matrices to make it ease to understand the transformed matrix. The null matrix in the upper right corner means that the matrices for Poisson's equation are to be unchanged. The lower left matrix, $-A_{\phi_n\psi}D_{\psi\psi}^{-1}$, is the key matrix of the MSP scheme. In contrast to the ABF scheme, where all the block matrices in T_{ABF} are diagonal, $-A_{\phi_n\psi}D_{\psi\psi}^{-1}$ has the same density as does $A_{\phi_n\psi}$.

The transformed matrix B can be written as follows:

$$B \equiv \begin{bmatrix} A_{\psi\psi} & D_{\psi\phi_n} \\ -A_{\phi_n\psi}D_{\psi\psi}^{-1}\tilde{A}_{\psi\psi} & A_{\phi_n\phi_n} - A_{\phi_n\psi}D_{\psi\psi}^{-1}D_{\psi\phi_n} \end{bmatrix} \quad (23)$$

where $\tilde{A}_{\psi\psi} \equiv A_{\psi\psi} - D_{\psi\psi}$ is a zero diagonal matrix. The null matrix in T_{MSP} guarantees that B has the same matrices as A for Poisson's equation, while the matrices of the current continuity equation have been changed. It is easy to see that $B_{\phi_n\psi}$ is $A_{\phi_n\psi}$ multiplied by a zero diagonal matrix $D_{\psi\psi}^{-1}(-\tilde{A}_{\psi\psi})$ in which all the elements are nonnegative.⁶

We can understand the performance of the MSP scheme by comparing the row sums of $A_{\phi_n\psi}$ and $B_{\phi_n\psi}$. Given a matrix $L = \{l_{ij}\}$, the row sum of its i th row, $s_{L,i}$, is

$$s_{L,i} = \sum_j |l_{ij}|. \quad (24)$$

Let A stand for $A_{\phi_n\psi}$ and C stand for $D_{\psi\psi}^{-1}(-\tilde{A}_{\psi\psi})$, we have

$$s_{A,i} = \sum_j |a_{ij}| \quad (25)$$

and

$$s_{C,i} = \sum_j |c_{ij}| \equiv \sum_j c_{ij} \quad (26)$$

where the nonnegativeness of C has been used in (26). Now, let B stand for $B_{\phi_n\psi}$, we have

$$s_{B,i} = \sum_{j,k} |a_{ij}c_{jk}| = \sum_j \left(|a_{ij}| \sum_k c_{jk} \right) = \sum_j (|a_{ij}| s_{C,j}). \quad (27)$$

It is clear that the row sums of $B_{\phi_n\psi}$ will be less than those of $A_{\phi_n\psi}$ if the row sums of $D_{\psi\psi}^{-1}(-\tilde{A}_{\psi\psi})$ are strictly less than one. This is indeed the case since $A_{\psi\psi}$ is a strictly diagonally dominant M matrix. Furthermore, $s_{C,j}$ is substantially less than one if the carrier terms dominate $D_{\psi\psi,j}$. This is because $s_{C,j}$ is no bigger than the ratio between the contribution of the Laplace operator to the diagonal, $(D_{\psi\psi,j}^{\text{Lap}} \equiv V_j^{-1} \sum_k \epsilon_{jk} A_{jk}/d_{jk})$, and the diagonal itself ($D_{\psi\psi,j} \equiv n_j + p_j + D_{\psi\psi,j}^{\text{Lap}}$). We believe that this strict reduction in the row sums of $B_{\phi_n\psi}$ with respect to those of $A_{\phi_n\psi}$ provides the mathematical basis for the performance of the MSP scheme. In addition, with the explicit block matrix formulation, the MSP scheme can also be used with the block iteration approach.⁷ In fact, both MSP and ABF based block-iterative Newton iterations are implemented in the solver. Further discussion is deferred to a future time.

From the above analysis, we can understand the reasons for the deterioration of MSP's performance as V_{ds} increases. A closer look at $A_{\phi_n\psi}$ reveals that the magnitude of its entries is proportional to the difference of Φ_n between two neighboring nodes. Therefore, $A_{\phi_n\psi}$ grows as V_{ds} increases. Furthermore, since an increase in V_{ds} causes the space-charge region to include more nodes, the row sums of $D_{\psi\psi}^{-1}(-\tilde{A}_{\psi\psi})$ corresponding to these newly de-

⁶Since $A_{\psi\psi}$ is a M matrix, all its off-diagonal elements, which constitute $\tilde{A}_{\psi\psi}$, are non-positive. Thus, all the elements of $(-\tilde{A}_{\psi\psi})$ are nonnegative. The diagonal elements of $D_{\psi\psi}^{-1}$ are strictly positive as the result of the diagonal elements of $A_{\psi\psi}$ being strictly positive.

⁷The fact that $B_{\phi_n\psi}$ is more dense than $A_{\phi_n\psi}$ is not a problem. Since it is only used in the residual update, we can use $(-A_{\phi_n\psi}D_{\psi\psi}^{-1}(\tilde{A}_{\psi\psi}\delta\psi))$ to realize $B_{\phi_n\psi}\delta\psi$.

pleted nodes will increase since the carrier terms no longer dominate $D_{\psi\psi}$ there. As a result of these two factors, $B_{\psi\psi}$ increases with increasing V_{ds} and this increase causes the deterioration in the performance of the MSP scheme. As another example for the utility of this analysis, we analyze the reason for the instability of the original SP scheme. There is only one difference between T_{MSP} and T_{SP} , the transformation matrix for the SP scheme. Specifically, $D_{\psi\psi}$ in T_{MSP} is replaced by $D_{\psi\psi}^{SP} \equiv D_{\psi\psi} - D_{\psi\psi}^{LSP}$ in T_{SP} . With the reduced denominator, some row sums of the corresponding $B_{\psi\psi}$ now may become larger than those in $A_{\psi\psi}$. For a node j in the space-charge region, $D_{\psi\psi}^{SP}$ may be much smaller than $D_{\psi\psi}^{LSP}$, and the increase in the row sum may be very significant. We believe that this is the source for the observed instability of the original SP scheme.

IX. CONCLUSION

In this paper, we have presented a parallel 3-D one-carrier device solver which shows the promise of handling practical device problems at an acceptable CPU cost. Slotboom variables are used in the current continuity equation to exploit the efficiency provided by the iterative solution methods for symmetric matrices. The Scharfetter-Gummel formula is used with the Slotboom variables to provide more accurate current estimation. By using appropriate scaling schemes, we have alleviated the problem of variable overflow and thereby produce a feasible bias range adequate for most applications. We also proposed a scheme to produce n, p variables as the scaled Slotboom variables which enables the use of acceleration schemes like the MSP scheme with the n, p variables.

Based on physical understanding, we have proposed a simple yet effective scheme to estimate the damping coefficients in the damped Newton scheme for Poisson's equation. We have improved a previously proposed initial guess scheme by using a novel update procedure during the Poisson solution stage after the initial guess step which eliminates the nonphysical carrier concentration overshoot at large bias steps. With the improved robustness, the device solver achieved convergence for initial biases of the full 5 V used in most VLSI applications.

Again, based on physical insight, we have proposed a modification to the singular perturbation scheme for accelerating the traditional Gummel iteration. We have demonstrated the success of the MSP scheme by several initial bias calculations for a MOSFET simulation. Finally, we have presented block matrix analysis of the MSP scheme which gives insight into the scheme's performance and allows it to be used in block-iterative Newton iterations.

APPENDIX

Using the quasi-Fermi level and the Slotboom variable (ϕ_n, Φ_n), electron current density can be written as

$$\begin{aligned}\bar{J}_n &= -q\mu_n n \nabla \phi_n \\ &= -q\mu_n \exp(\psi - \phi_n) \nabla \phi_n \\ &= q\mu_n \exp(\psi) \nabla \Phi_n.\end{aligned}\quad (A.1)$$

In the x direction, (A.1) can be rewritten as

$$\frac{J_{nx} \exp(-\psi)}{q\mu_n} = \frac{\partial \Phi_n}{\partial x}. \quad (A.2)$$

Integrating (A.2) from x_i to $x_{i+1} \equiv x_i + h_i$, we get

$$\frac{1}{q} \int_{x_i}^{x_{i+1}} \frac{J_{nx}(x) \exp(-\psi(x))}{\mu_n(x)} dx = \Phi_n(x_{i+1}) - \Phi_n(x_i). \quad (A.3)$$

We now consider the quadrature of integral in (A.3). The rectangle rule [21] gives the following expression

$$\begin{aligned}&\int_{x_i}^{x_{i+1}} \frac{J_{nx}(x) \exp(-\psi(x))}{\mu_n(x)} dx \\ &= h_i \left(\frac{J_{nx}(x_{i+1/2}) \exp(-\psi(x_{i+1/2}))}{\mu_n(x_{i+1/2})} + O(h_i^2) \right)\end{aligned}\quad (A.4)$$

where $x_{i+1/2} \equiv (x_i + x_{i+1})/2$. Equation (A.4) indicates a discretization accuracy of the order of 2 for $J_{nx}(x_{i+1/2})$ if both $\exp(-\psi(x_{i+1/2}))$ and $\mu_n(x_{i+1/2})$ have the same order of discretization accuracy. The order of two accuracy for $\mu_n(x_{i+1/2})$ can be easily achieved by averaging their values at x_i and x_{i+1} . As shown by Yu [22], the S-G scheme amounts to use $(\exp(-\psi(x_i)) - \exp(-\psi(x_{i+1}))) / (\psi(x_{i+1}) - \psi(x_i))$ to approximate $\exp(-\psi(x_{i+1/2}))$. Using the Taylor expansion of $\exp(-\psi(x_i))$ and $\exp(-\psi(x_{i+1}))$ with $x_{i+1/2}$ as the origin, it can be shown that this approximation also has a discretization accuracy of the order of two. In summary, the S-G formulation can be obtained without the assumptions of constant current density along the interval $[x_i, x_{i+1}]$ and it has a discretization accuracy of the order of 2.

ACKNOWLEDGMENT

The authors wish to thank Conor Rafferty for valuable discussions. They also wish to thank Dr. Arthur Raefsky for suggesting the use of the GMRES algorithm.

REFERENCES

- [1] R. F. Lucas, K. Wu, and R. W. Dutton, "A parallel 3-D poisson solver on a hypercube multiprocessor," in *IEEE Int. Conf. Computer Aided Design*, pp. 442-445, 1987.
- [2] J. A. Greenfield, S. E. Hansen, and R. W. Dutton, "Two dimensional analysis for device modeling," in Tech. Report G-201-7, Stanford Electronics Lab., Stanford Univ., Stanford, CA, 1980.
- [3] R. E. Bank, D. J. Rose, and W. Fichtner, "Numerical methods for semiconductor device simulation," *IEEE Trans. Electron Devices*, vol. ED-30, pp. 1031-1041, Sept. 1983.
- [4] A. F. Franz, G. A. Franz, S. Selberherr, C. Ringhofer, and P. Markowich, "Finite boxes—A generalization of the finite-difference method suitable for semiconductor device simulation," *IEEE Trans. Electron Devices*, vol. ED-30, no. 9, pp. 1070-1082, Sept. 1983.
- [5] C. den Heijer, "Preconditioned iterative methods for nonsymmetric linear systems," in *Proc. Int. Conf. on Simulation of Semiconductor Devices and Processes* (Ed. D. R. J. Owen), Pineridge Press, Swansea, U.K., June 1984.
- [6] Y. Saad and M. H. Shultz, "GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems," *SIAM J. Sci. Stat. Comp.*, vol. 7, pp. 856-869, 1986.

A Methodology for Parallelizing PDE Solvers: Application to Semiconductor Device Simulation^{*}

Bruce P. Herndon[†] N.R. Aluru[†] Arthur Raefsky[†]
Ronald J.G. Goossens[‡] Kincho Law[†] Robert W. Dutton[†]

Abstract

A methodology for creating parallel grid-based partial differential equation (PDE) solvers from serial ones with minimal changes to the existing code and data structures is presented. The approach is based upon the single-program-multiple-data programming model and is applicable to a wide range of PDE solution techniques while remaining easily portable to most message-passing distributed-memory parallel computers. Results of applying this methodology to semiconductor device simulators are given.

1 Introduction

The complexity of synchronization and communication makes the development of parallel applications far more Byzantine than the development of serial ones. One way to simplify parallel application development is to use existing serial applications as building blocks without modifying the original code or data structures. By exploiting the development and validation history of the serial code, development can concentrate on the parallel aspects of the application. Our methodology employs the single-program-multiple-data (SPMD) to create parallel grid-based partial differential equation (PDE) solvers from serial ones with minimal changes to the existing code and data structures.

2 Software Model

The SPMD model provides a flexible and intuitive framework for parallelization of existing grid-based PDE solvers without forcing significant changes to either the data structures or the code. The serial code is encapsulated and strictly controlled by a choreographer which coordinates communication and synchronization in the parallel environment. Problems are decomposed using well-known grid decomposition techniques. Each processor of the parallel machine solves one section of the partitioned grid. Data dependencies which exist along the boundaries between partitions are satisfied by the choreographer through inter-processor communication and a global solution identical to the serial solution is computed.

Most nonlinear PDE solvers contain the following code modules: user interface (UI), nonlinear solver, physical model evaluation, matrix formation, and matrix solution. The UI parses input files, performs file I/O functions, and displays computational results. The remainder of the code solves the PDEs by employing an appropriate discretization and solution method. Generally, the UI routines take negligible time and are inherently serial. In order to accommodate this dichotomy, we split the serial application into two separate programs: a front-end program to handle the serial UI tasks and a parallel program containing the PDE solver. Our choreography model accommodates this natural decomposition through a two-stage choreographer. A coarse-grain choreographer partitions the global simulation grid and controls the data motion between the UI and the parallel PDE solver. Aside from the code to manage data structures unique to each PDE solver, the bulk of the coarse-grain choreography code can be reused without significant modification. This includes the domain decomposition and communication primitives between the two programs. A fine-

^{*} Research funded by the State of CA under contract C90-072 and ARPA under contract DAA100391-C-0043.

[†] Integrated Circuits Laboratory, Stanford University, Stanford, CA, 94305-4055.

[‡] National Semiconductor Corporation, Santa Clara, CA, 95052-8090.

grain choreographer provides synchronization and communication for the parallel solution and is application-specific due to each PDE solver's distinctive communication and synchronization patterns.

3 Semiconductor Device Applications

As semiconductor manufacturing technology becomes increasingly complex and costly, device simulators become crucial in quantifying the electrical behavior of devices. The interplay between adjacent devices as they are scaled to submicron sizes becomes more important and can dramatically increase the memory and execution time requirements of simulations. Device simulators must realize significant performance gains to provide reasonable turnaround for device and process designers. We have applied our parallelization methodology to two very different semiconductor device simulators. One code exemplifies "dusty-deck" PDE solvers using fully implicit direct solution techniques. The other is typical of modern finite element software and utilizes implicit iterative solution methods.

The simulator PISCES [2] is a well-known and widely-used two-dimensional, two-carrier device modelling program. It solves the traditional drift-diffusion equations using a finite volume formulation on a non-uniform triangular grid. A direct linear solver is employed due to the extremely ill-conditioned matrices arising from the discretization method. The coarse-grain choreographer manages data motion by transferring FORTRAN COMMON blocks between the UI and PDE solver. Non-local data references exist in the linear solver, nonlinear solver, and physical models. The fine-grain choreographer manages all parallel communication and synchronization external to the original code modules. This minimizes the changes to the original serial code and preserves its value.

The same methodology was then applied to a second device modelling program. The simulator, FIESTA-HD [1], is currently being developed to solve the state-of-the-art hydrodynamic device equations employing a space-time Galerkin/Least-Squares finite element method. FIESTA-HD was built upon an existing finite element solver for compressible Euler and the Navier-Stokes equations. The linear solvers currently employed are GMRES to solve the non-symmetric hydrodynamic equations and conjugate-gradient for the symmetric Poisson equation. The most time-consuming parallelization task was customizing the coarse-grain choreographer for the application's data structures. The remainder of the work involved adapting the iterative linear solvers for parallel execution. The parallel communication was limited entirely to the linear solvers. As with PISCES, modifications to the existing serial code and data structures were minimal.

4 Results

Initial parallel development took place on the 32-node Intel iPSC/860 in our lab. Once the initial parallel codes were finished, we ported PISCES to the Thinking Machines CM-5 and the IBM SP-1 and FIESTA-HD to the Intel Touchstone Delta and the IBM SP-1. These implementations were easily managed and demonstrate portability of applications developed using our methodology.

As a demonstration of the utility of our parallel codes, we have run simulations using increasingly large and complex realistic device structures. We simulated the structures using an IBM RS/6000 Model 560 and the parallel architectures. As grids scaled to modest and large sizes, the parallel codes performed significantly better than the workstation versions. We routinely achieved more than order-of-magnitude reductions in wall-clock execution time for moderately-sized grids using moderately-sized 16-node and 32-node parallel computers. For example, simulating a CMOS inverter with roughly 11,000 grid points using PISCES required 4.4 hours on the workstation and 0.3 hours on our 16-node SP-1. Moreover, using these parallel machines, we were able to solve ultra-large structures for which a serial solution could not be obtained due to resource constraints. Through the application of our parallelization methodology, we believe any PDE solver with a similar program structure could be adapted to provide improved solution capabilities.

References

- [1] N. Aluru, A. Raefsky, P. Pinsky, K. Law, R. Goossens, and R. Dutton, "Finite element formulation for hydrodynamic semiconductor device equations," *Comp. Meth. Appl. Eng.*, v.107, pp. 269-298, 1993.
- [2] M. Pinto, C. Rafferty, and R. Dutton, "PISCES-II: Poisson and continuity equation solver," Stanford Electronics Lab, Stanford Univ., Stanford, CA, 1985.

PISCES MP - Adaptation of a Dusty Deck for Multiprocessing

Bruce P. Herndon, Arthur Raefsky, and Ronald J.G. Goossens
 Integrated Circuits Laboratory, Stanford University, Stanford, CA 94305

I. INTRODUCTION

Scalable multiprocessors offer high performance with relatively low cost. Unfortunately, the programming model required to take advantage of these architectures is a radical departure from traditional paradigms. Most users are unwilling to discard the knowledge and expertise captured by existing dusty-deck programs in exchange for a faster yet unproved and unfamiliar parallel code. To explore the potential for providing vastly improved dusty-deck performance while preserving the knowledge implicit in the program, we have parallelized the device simulator PISCES [1] on an Intel iPSC/860™ hypercube.

Section II gives a brief overview of PISCES. Section III describes the methods used to transform PISCES into a parallel code. A demonstration of the computational power of the new parallel device solver is presented in Section IV. Improvements to the linear solver are discussed in Section V. Finally, conclusions are given in Section VI.

II. Overview of PISCES

PISCES is a two-dimensional device simulator consisting of approximately 40,000 lines of FORTRAN-77. Code development has been ongoing throughout the last ten years, involving several generations of graduate students and researchers. Although the program structure is rather inelegant, great care has been taken to validate the code as well as to improve and calibrate the physical models. It solves Poisson's equation and the continuity equations below:

$$\nabla(\epsilon \nabla \Psi) = -q(p - n + N_D^+ - N_A^-) - \rho_f$$

$$\frac{\partial n}{\partial t} = \frac{1}{q} \nabla \cdot J_n - U_n$$

$$\frac{\partial p}{\partial t} = \frac{1}{q} \nabla \cdot J_p - U_p$$

The equations are discretized using irregular triangular grid and are solved using either Newton or Gummel nonlinear schemes. A large number of physical models are supported. The sparse systems of linear equations arising from these methods are solved using an optimized sparse direct solver as described in [2]. Figure 1 shows the major code components. Lucas observed in [3] that for even small simulation grids, PISCES spends between 77% and 96% of its runtime solving the coupled nonlinear device equations. The nonlinear solver

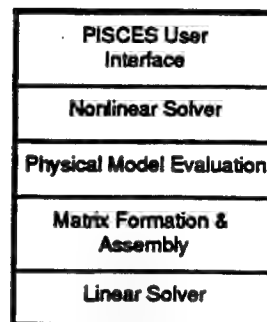


Figure 1
Major elements in PISCES

repeatedly forms element matrices, assembles a global matrix, solves the resulting sparse system, and updates the nonlinear solution. Recent experience shows nonlinear solution times grow to be more than 99% of the runtime for moderate to large grid sizes. The remaining fraction of time is spent in the user interface (UI) parsing user input, performing I/O, and generating grid.

III. Parallelization of PISCES

Typical PISCES simulations require several hours on moderate grid sizes and days on large grids. Clearly, significant performance gains would be welcomed by users. Restructuring the nonlinear solver and all of its requisite routines to run in parallel would breathe new life into the simulator. However, the UI is inherently serial and must be treated differently. In order to accommodate this dichotomy, we split the code into two programs. Figure 2 shows the structure of PISCES MP. The bulk of PISCES MP runs on the hypercube including all code for nonlinear solution, model evaluation, matrix formation, matrix assembly, and linear solution. Although we left the majority of PISCES code untouched, many changes were necessary. Fortunately, changes rarely pervaded the entire code. For instance, we were forced to add data structures to map each processor's local domain into the global simulation grid and to determine each processor's responsibility for shared portions in each domain. We were also forced to modify those physical models and assembly routines that relied on non-local information. For example, all grid points attached to an electrode must be given a consistent potential value. If these grid points are distributed across multiple processors, the processors must communicate to determine the proper value. Finally, we replaced the linear

direct method, each processor directly eliminates all local equations and updates a dense block corresponding to the shared equations. Rather than solve the dense shared block directly, we use the preconditioned generalized minimal residual (GMRES) algorithm [5]. GMRES requires less global data traffic than a direct method. Table 2 compares the linear solution times on the 9200 grid point example described earlier. The solution times for a single linear solution are given. This simulation required in excess of 120 linear solutions. Not surprisingly, the fully direct method is faster for a small number of processors due to the large amount of local computation coupled with the small amount of necessary data transfer. As expected, for larger numbers of processors the hybrid method outperforms the fully direct method by reducing the amount of shared data transfer. This allows for the exploitation of greater concurrency and results in faster overall solution times.

	Hybrid	Direct
Computational Unit	Time (s)	Time (s)
iPSC/860 4 CPU	11.023	9.604
iPSC/860 8 CPU	6.821	5.618
iPSC/860 16 CPU	3.942	4.819
iPSC/860 32 CPU	3.768	6.951

Table 2
*Comparison of linear solution times on
9200 grid point example*

VI. Conclusions

In this paper, we have described the parallelization of PISCES. We have retained the valuable expertise captured in the long-term development of the program. Our initial results show significant performance gains. In fact, the program not only runs existing simulations faster but also provides the capability of solving vastly larger problems than originally feasible. We have also addressed the communication bottleneck created by the direct solver when using large numbers of processors. We have implemented a hybrid solver that produces greater parallel efficiency in these cases.

Acknowledgments

We would like to thank Horst Simon for providing the spectral nested dissection code and Bob Lucas for improving our understanding and implementation of the multi-frontal method. This research was sponsored by the State of California's Department of Commerce under grant number C90-072 and by DARPA.

References

- [1] M.R. Pinto, C.S. Rafferty, H.R. Yeager, and R.W. Dutton, "PISCES-IIB Supplementary Report," Stanford Electronics Lab., Stanford Univ., Stanford, CA, 1985.
- [2] D.A. Calahan and P.G. Buning, "Vectorized General Sparsity Algorithms with Backing Store," SEL Report 96, Systems Engineering Laboratory, University of Michigan, Ann Arbor, MI, 1977.

- [3] R.F. Lucas, "Solving Planar Systems of Equations on Distributed-Memory Multiprocessors," Ph.D. Dissertation, Stanford Univ., Dec., 1987.
- [4] A. Pothen, H.D. Simon, and K.P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," SIAM J. Mat. Anal. Appl., 11 (1990), pp. 430-452.
- [5] Y. Saad and M.H. Shultz, "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems," SIAM J. Sci. Stat. Comp., vol. 7, pp. 856-869, 1986.

Abstract

This paper presents a methodology for adapting PDE solvers for parallel execution based upon the single-program-multiple-data programming paradigm. Our approach minimizes changes to existing code and data structures, thereby preserving the value captured within "dusty-deck" programs. The resulting parallel application is easily portable to most message-passing distributed-memory architectures. To demonstrate the viability of our methodology, the commercially-available, semiconductor modelling program, PISCES, has been adapted for parallel execution. Our experience shows that all non-local references can be resolved through careful choreography without extensive modifications to the original code. The parallel simulator currently runs on the Intel iPSC/860 and the Thinking Machines CM-5. Simulating realistic complex device structures, we have achieved remarkable performance gains over high-performance serial workstations. We also demonstrate the ability, due to the scalability of the parallel simulator, to simulate structures too large for our existing serial computers. This simulation capability could provide immeasurable benefits in the competitive semiconductor industry.

A Methodology for Parallelizing PDE Solvers: Application to PISCES

Bruce P. Herndon

Integrated Circuits Laboratory, Stanford University
231A Applied Electronics Lab
Stanford University, Stanford, CA 94305-4055
Tel: (415) 723-1482, Fax: (415) 725-7298
e-mail: herndon@gloworm.stanford.edu

Arthur Raefsky

Centric Engineering Systems
3801 E. Bayshore Rd.
Palo Alto, CA 94303
Tel: (415) 960-3600, Fax: (415) 940-1252
e-mail: raefsky@centric.com

Ronald J.G. Goossens

National Semiconductor Corporation
P.O. Box 58090
Santa Clara, CA 95052-8090
Tel: (408) 721-2420, Fax: (408) 721-7266
ronald@ammon.nsc.com

Robert W. Dutton

Integrated Circuits Laboratory, Stanford University
203 Applied Electronics Lab
Stanford University, Stanford, CA 94305-4055
Tel: (415) 723-4138, Fax: (415) 725-7298
e-mail: dutton@gloworm.stanford.edu

submitted to: Journal of Computer & Software Engineering,
Special Issue on Parallel Computing

1. Introduction

As time-to-market for integrated circuits diminishes and manufacturing technology becomes increasingly complex and costly, semiconductor device simulators become crucial in quantifying the electrical behavior of devices. Simulators are now used to perform not only design verification but also manufacturability and scalability studies. The interplay between adjacent devices as they are scaled to submicron sizes becomes more important and can dramatically increase the memory and execution time requirements of simulations. These two trends mean device simulators must realize significant performance gains to provide reasonable turnaround for device and process designers. The use of scalable, high-performance parallel architectures to deliver that performance will become strategically important in the continued success of large-scale device simulation.

In recent years, distributed-memory parallel computers built with high-performance microprocessors and supporting the message passing paradigm have become the standard commercial parallel architecture. Most commercial vendors (e.g., IBM, Cray, Intel, and Thinking Machines) produce machines of this type. Likewise, the single-program-multiple-data (SPMD) parallel programming model [1] has become a popular paradigm for developing parallel applications for this class of machines. Emergence of a single hardware abstraction and its requisite programming model should portend the migration of parallel machines from research laboratories to industrial production environments. Unfortunately, one key piece of the puzzle is missing: *industrial applications*. Although many researchers have developed codes for parallel architectures, commercial software development has been slow because the programming model required to take advantage of these architectures is a radical departure from traditional paradigms. Additionally, most commercial users are unwilling to discard the knowledge and expertise captured by their existing “dusty-deck” programs in exchange for a faster yet unproved and unfamiliar parallel code. One way to bootstrap parallel application development is to port existing, well accepted industrial applications to parallel machines while retaining the original code and data structures without major modifications. A method of parallelization that preserves knowledge inherent in “dusty-decks” pleases users by maintaining a familiar environment while providing an improved solution capability. This approach also aids parallel software and hardware designers by rapidly exposing issues involved with running full-scale industrial applications on parallel machines. In this paper we present a methodology for creating parallel grid-based partial differential equation (PDE) solvers from serial ones with minimal changes to the existing code and data structures. Our

method is based upon the SPMD programming model and is easily portable to most message-passing distributed-memory machines. To verify our method and to illustrate the potential for providing vastly improved “dusty-deck” performance, we have adapted a commercially-supported and industry-standard device simulator, PISCES [2,3], for parallel execution. The code currently runs on two distributed-memory architectures: the Intel iPSC/860 and the Thinking Machines CM-5.

An outline of this paper follows: After a brief description of the serial semiconductor device simulator, we present in Section 3 our methodology for adapting PDE solvers for parallel execution accompanied by a discussion of our experiences adapting PISCES for parallel execution. In Section 4, simulation results from a series of increasingly complex grids defining a large, multi-device structure illustrate the utility of the parallel application. Finally, conclusions are drawn in Section 5.

2. Overview of PISCES

PISCES is a large and complex application which solves problems strategic to the semiconductor industry. The code encompasses many areas of research in both physics and computational mathematics. It is a two-dimensional, two-carrier semiconductor device modeling program developed primarily at Stanford University over the past fifteen years and is well known throughout the semiconductor industry. The program is available from Stanford or from one of several Technology CAD vendors who support the code commercially. To date, there are more than one thousand industrial and academic users. PISCES solves Poisson’s equation and the semiconductor continuity equations below:

$$\nabla (\epsilon \nabla \Psi) = -q(p - n + N_D^+ - N_A^-) - \rho_F \quad \text{Poisson's equation}$$

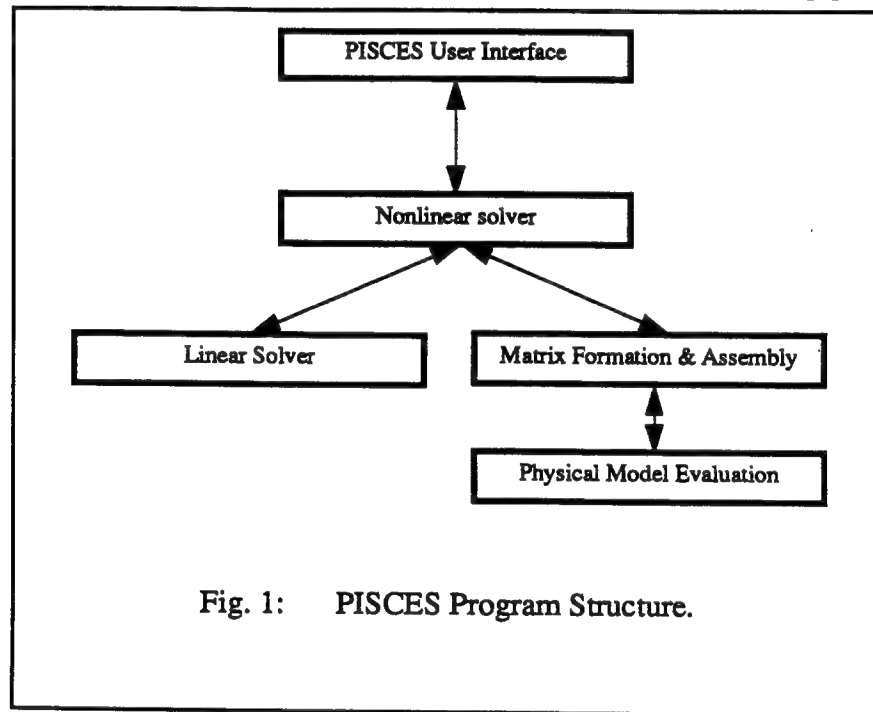
$$\frac{\partial n}{\partial t} = \frac{1}{q} \nabla \bullet J_n - U_n \quad \text{Electron Continuity}$$

$$\frac{\partial p}{\partial t} = \frac{1}{q} \nabla \bullet J_p - U_p \quad \text{Hole Continuity}$$

Ψ is the electric potential, and n and p are the electron and hole concentrations. N_D^+ and N_A^- are the ionized impurity concentrations and ρ_F is the fixed-charge density. J_n and J_p are the electron and hole current densities and U_n and U_p are the electron and hole recombination rates.

These coupled nonlinear PDEs are discretized using a finite volume formulation on a non-

uniform triangular grid. The resulting algebraic equations are solved using a nonlinear iteration method. The coarse-grain structure of the simulator is shown in Figure 1. This program structure is typical of many grid-based nonlinear PDE solvers. The user interface (UI) parses input files, performs file I/O functions, and displays simulation results via plotting utilities. The nonlinear solver supports a wide range of solution techniques and contains both a fully-coupled Newton [4] solver and staggered Gummel [5] scheme. PISCES offers a large and diverse set of physical models. Improvements and additions to the physical models are the most active areas in the continuing development of the simulator. During the nonlinear solution process, a sparse linear system is assembled in a triangle-by-triangle fashion using a subset of the available physical models. It has been shown in [6] that the matrices arising in this type of device simulation are extremely ill-conditioned and not easily solved using iterative techniques. Therefore, the sparse system of linear equations is solved using an optimized sparse direct solver as described in [7].



An example PISCES simulation grid is shown in Figure 2. The structure represents the 2D cross-section of a CMOS inverter. This coarse grid contains roughly 1,600 nodes and 3,000 triangles. The structure was created using standard one micron layout rules and was derived from the model of an SRAM cell produced by our lab in conjunction with an industrial partner. We ran a very simple simulation to compute the DC terminal characteristics using a finer mesh than shown (roughly 8,000 nodes and 15,000 triangles) for improved accuracy. After reading the grid and set-

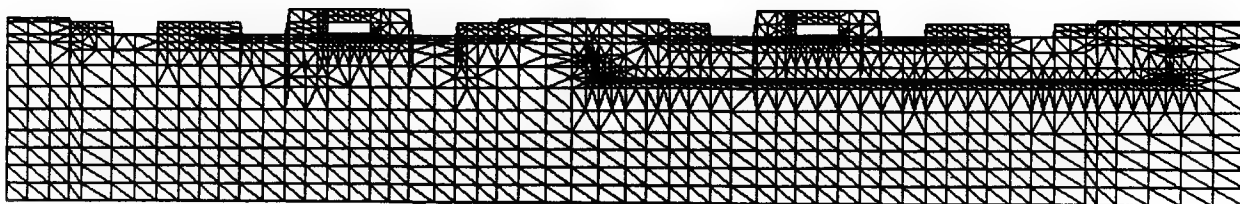


Fig. 2: Grid structure of a CMOS inverter.

ting the proper parameters, the simulation brings the power supply to 5V and then ramps the input gates from 0V to 5V. The output is shown in Figure 3. As expected in an inverter circuit, the output experiences a rapid transition from 5V to 0V as the input voltage increases.

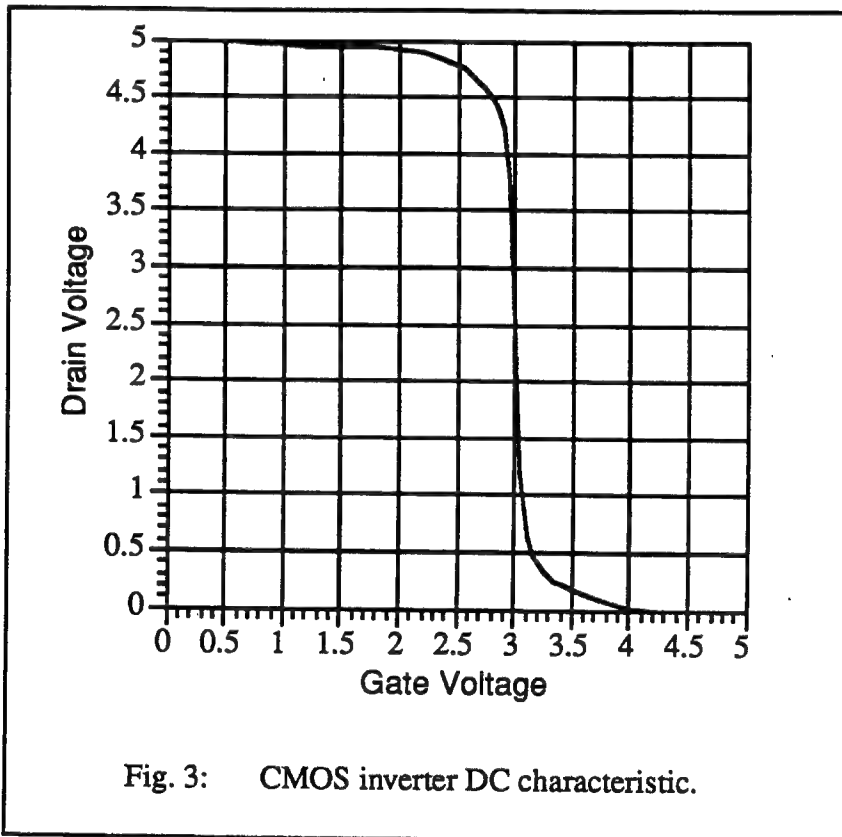


Fig. 3: CMOS inverter DC characteristic.

This simple simulation required more than two hours on an IBM RS/6000 Model 560F, the fastest serial computer available in our lab. Using a slightly finer mesh containing nearly 11,000 nodes and 21,000 triangles, the simulation required more than four hours. Simulations to perform more detailed analyses of this structure can easily take several days to complete. If we

were to perform multiple simulations as part of a manufacturability study, several weeks would be needed. A significant boost in the performance of PISCES will be needed if it is to keep pace with current and future demands.

3. A Recipe for Parallelization of PDE Solvers

Before blindly initiating the development of any parallel application, one should estimate the expected benefits and determine if the expected effort will be rewarded. We know that parallel grid-based PDE solvers have had measurable success [8-11]. However, the current version of PISCES consists of approximately 40,000 lines of FORTRAN-77. During its fifteen-year life-span several generations of researchers have modified the code. Due to this continual development, the code has become unwieldy.

Despite its inelegant program structure, great care has been taken to validate the code as well as to improve and calibrate the physical models. In fact, industrial users have invested considerable effort calibrating PISCES with experimental data to further improve accuracy. Moreover, its long lifetime has allowed a large, experienced, and satisfied user base to be established. These users understand the subtleties of the simulator and have developed sophisticated solution strategies unique to PISCES. Finally, both the long lifetime and large user base mean that many bugs and inconsistencies in the code have been exposed and either corrected or worked around. Developing a new application would force users and developers to repeat an enormous amount of non-trivial and time-consuming work beyond initial code development. These facts make strong arguments for parallelizing the current code provided that the effort is not too great and the parallel code maintains consistency with the original serial code. This requires that changes to the existing code be minimized and localized. We can not fundamentally alter either the data structures or the algorithms if the value of the code is to be preserved.

To insure that the parallelization can be done quickly, smoothly, and without major changes to the program, it is crucial to select an amenable parallel programming model. The two dominant parallel programming paradigms are data-parallel [12] and SPMD. Data-parallel compilers require the parallelism to be made explicit in the data structures. Modifying the data structures in PISCES to be compliant with this model would require significant recoding, precluding a data-parallel implementation if our original development goals are to be met. On the other hand, the SPMD model appears to provide a flexible and intuitive framework for parallelization of grid-based PDE solvers without forcing significant changes to either the data structures or the code. Using well-known grid decomposition techniques [13-17], the simulation grid can be broken into

disjoint pieces. Each processor of the parallel machine can run a copy of the device simulator to solve one section of the partitioned grid. Data dependencies will exist along the shared boundaries between partitions. If these data dependencies can be satisfied through inter-processor communication, a global solution identical to the serial solution can be computed. This approach allows the bulk of the code and data structures to remain unchanged. Using this technique, we have parallelized PISCES for distributed-memory computers. The Intel iPSC/860 hypercube in our lab was used for the initial development. Once the parallel application was completed, we moved the code to the Thinking Machines CM-5 to verify portability.

All of the work described below required roughly eight months to complete. We feel that is a reasonable amount of time to preserve a code with a fifteen-year history while giving it a vastly improved solution capability. A step-by-step review of our scheme for porting grid-based PDE codes using the SPMD model and our experiences adapting PISCES using the model is given in the following sections.

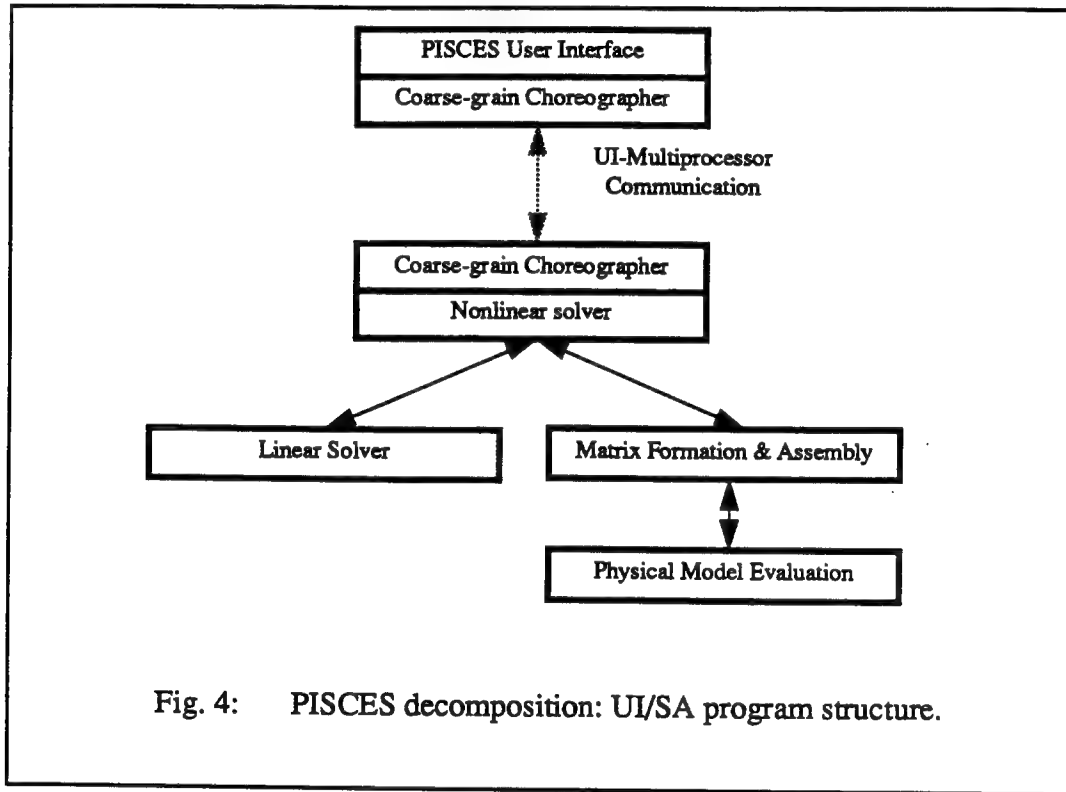
3.1 Deciding what to parallelize.

It was observed by Lucas[18] that for small simulation grids (less than 2000 grid points) PISCES normally spends more than 95% of its runtime solving the coupled nonlinear device equations. The remaining fraction of time is spent in the UI. Within the nonlinear solver roughly half the time is spent computing linear solutions to obtain updates to the nonlinear system. The other half of the nonlinear solution time is spent creating initial guesses, forming and assembling the Jacobian matrices, evaluating physical models, and applying nonlinear updates. More recent analysis shows nonlinear solution times grow to be more than 99% of the total runtime for larger grid sizes. As grid sizes increase, the linear solver scales more poorly than other parts of the code; nevertheless, for structures containing tens of thousands of grid points, 15-20% of the runtime remains outside of the linear solver. Amdahl's Law [19] tells us that merely adding a parallel solver to PISCES would yield at best a 5x speedup for many realistic grid sizes.

Restructuring the nonlinear solver and all of its requisite routines to run in parallel is vital to extending the utility of the simulator. However, the UI routines take negligible time and are inherently serial. In order to accommodate this dichotomy, we propose splitting the application into a front-end program to handle the serial UI tasks and a separate parallel program containing the actual device solver.

3.2 Separating the serial code from the parallel code.

Before embarking upon the parallelization of the device solver, we propose to bifurcate PISCES into two serial programs: the serial UI program and a serial solver application (SA). This initial program decomposition resolves the data dependencies between the UI and serial SA, sets up a communication mechanism between the two, and creates a coarse-grain choreographer to coordinate their interaction. The resulting global structure of the simulator is shown in Figure 4. Since managing the hypercube already taxes the control processor of the iPSC/860, we opted to run the UI program on a remote SUN workstation. The serial SA was put on one node of the hypercube.



After splitting the code between the two programs, we identified all of the data structures required by the serial SA. Like most large FORTRAN-77 programs, PISCES data structures are defined and passed via COMMON blocks. It was a relatively simple task to identify all of the COMMON blocks needed by the serial SA. Preceding computation, the UI performs a wholesale transfer of necessary COMMON blocks to the serial SA using the communication routines described below.

To facilitate the data transfer, we created a communicational package using TCP/IP [20] to

connect the two programs. TCP/IP provides a very slow connection. (The data transfer rate is roughly 65kB/s in our network environment.) However, the communication package can easily be replaced when faster communication resources are available. The main complication involved accommodating the differing machine-word byte orderings of the Intel hypercube and the SUN workstation. In our communication routines, the receiver is responsible for byte swapping data upon receipt. To simplify this procedure, we separated the data into the five types used by PISCES: integer, real, double precision, character, and logical. Similarly typed data are grouped into large blocks for transmission between the UI and SA. This makes the swapping operation simpler and more efficient. Once properly received and reordered, data values are loaded into their corresponding COMMON blocks. In this fashion the data transfer is isolated from and transparent to the original code.

A coarse-grain choreographer to coordinate the transfer of data and solution commands was also necessary. The coarse-grain commands consist of a series of bias conditions (each bias condition requires a separate nonlinear solution) to be solved by the SA. The UI choreographer interfaces with the main dispatch loop of PISCES (the routine which parses user's commands and calls the appropriate subroutines) and intercepts commands that require remote processing. The UI choreographer then transfers the data necessary for the computation requested followed by the actual command. It then awaits notification from the SA choreographer that the command has been executed and receives the results. At the other end, the SA choreographer waits to receive messages, initiates the requested action, and returns the results. The changes to the original code were minimal since the choreographer is entirely external to both the UI and the SA. Only the dispatch loops were modified to interface with the choreographer.

3.3 Adding Domain Decomposition

We expose the parallelism in the problem through grid decomposition. Naturally, the first task for parallelization was to add a domain decomposition module (DDM) to the UI. At runtime, the user's grid structure will be given to the DDM which will then spread the grid among the processors. An example of grid-based decomposition is shown in Figure 5. The CMOS structure described earlier has been divided into sixteen partitions. The partitioning was created using the recursive spectral bisection (RSB) algorithm of Pothen *et. al.* [14]. The goal of the DDM is to divide the work equally among the processors while minimizing the communication by keeping boundaries small. We chose to use the RSB algorithm as the heart of our DDM. The DDM consists of the RSB code, a pre-processor to convert a PISCES grid description into an RSB grid

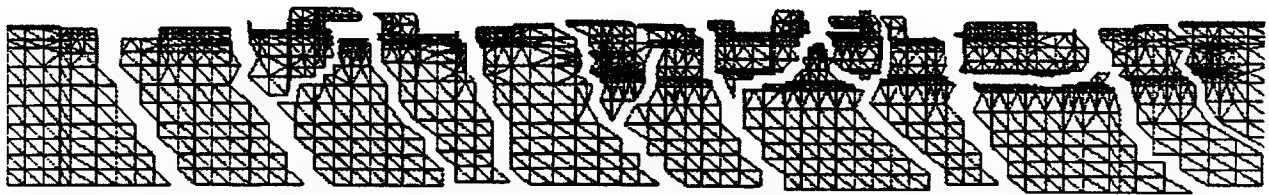


Fig. 5: 16-processor decomposition using Recursive Spectral Bisection.

description, and a post-processor to correct any inconsistencies in the partitioning. The changes to the UI code were minor: a call to the DDM after the grid structure is loaded produces the partitioning.

RSB was designed to handle homogenous grids (i.e., all nodes in the grid have equal characteristics). The addition of boundary conditions in the form of external electrodes (Figure 6) can

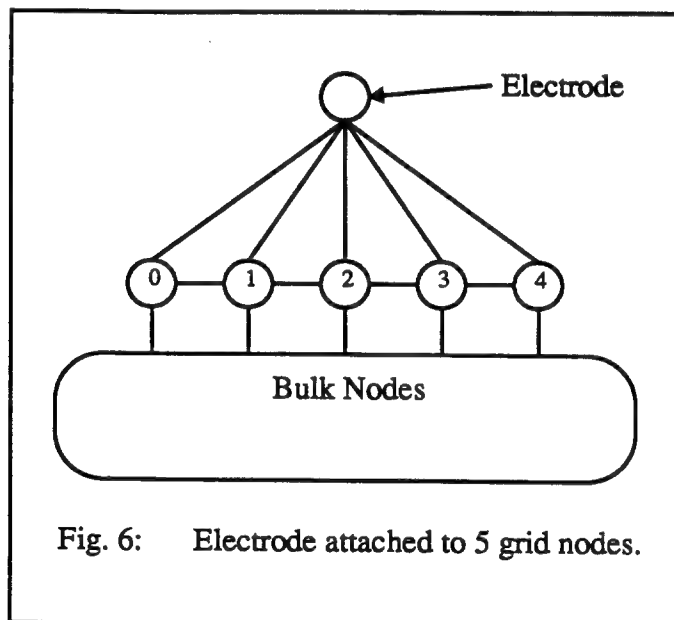
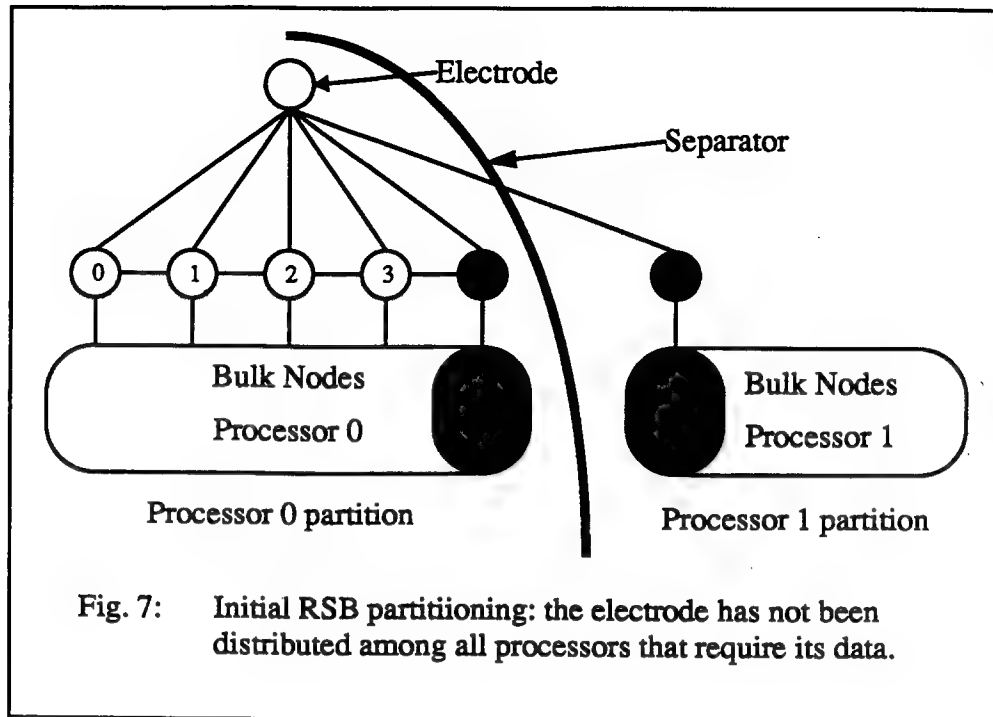
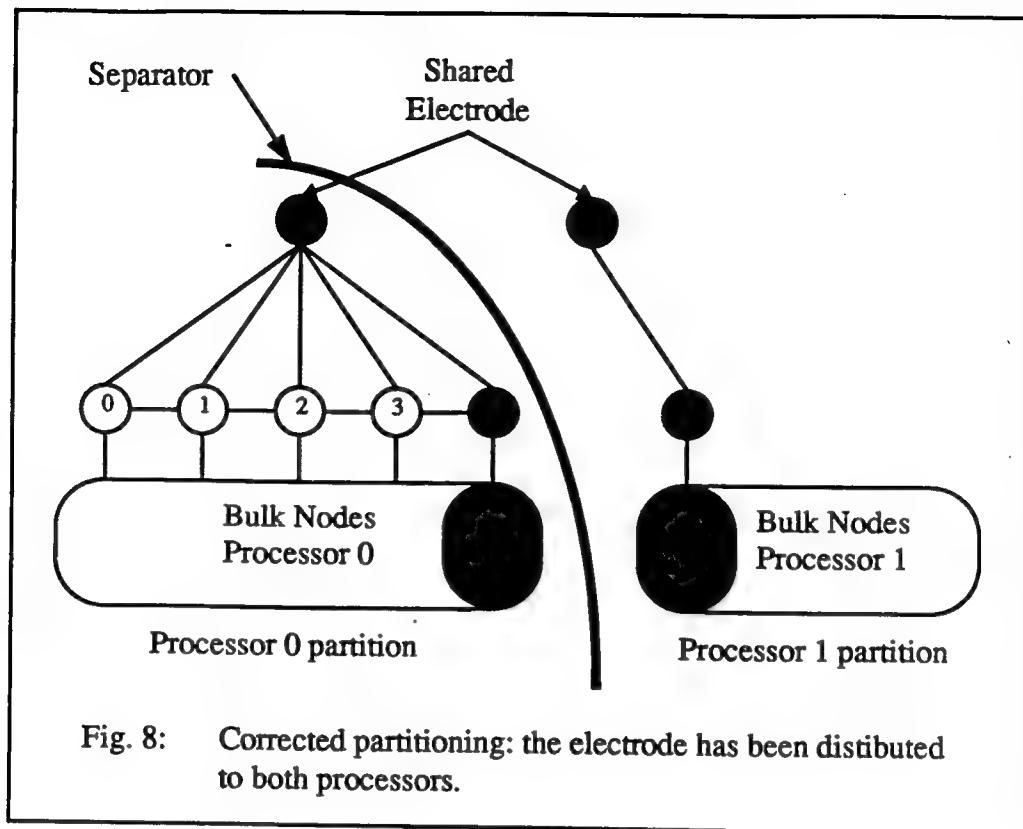


Fig. 6: Electrode attached to 5 grid nodes.

create inconsistencies which must be corrected by post-processing the RSB partitions. The electrodes are included in the grid description passed to the DDM; however, they are different from other nodes in the grid since they affect the assembly of all nodes connected to them. If the set of nodes attached to an electrode is spread across multiple processors by the domain decomposer (Figure 7), irregularities will occur if the electrode is not designated as shared among the processors. This can easily occur since the RSB cannot guarantee the an electrode will be divided among all partitions containing attached nodes. The post-processor examines the partitioning and insures



that all processors involved with nodes attached to an electrode are also involved with that electrode. The result of applying this operation causes electrodes to be shared across all involved processors. (Figure 8) and the boundary information to be distributed properly.

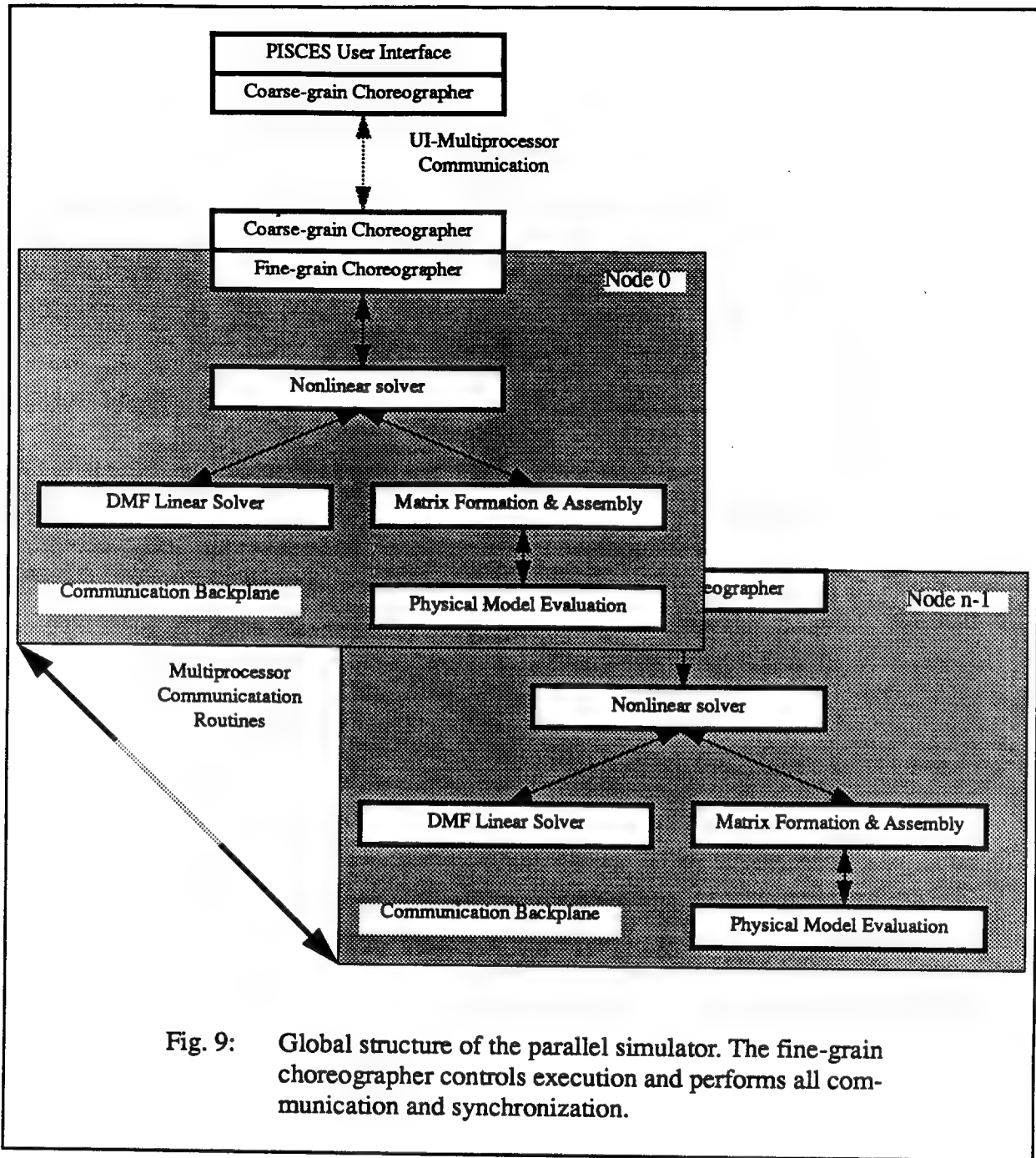


3.4 Parallel Data Transfer and Control

Once the UI has the ability to partition grids for parallel solution, it must be modified to send data to all processors in the parallel partition. Since it would be difficult to manage a separate TCP/IP connection to each processor (some machines have several hundred or more), the UI continues to connect to only one processor, *node_0*. Data to other nodes is sent through *node_0* and forwarded to the destination via the parallel message passing library. The Node Executive (NX), Intel's message passing library[21], is much faster than the TCP/IP connection making the extra hop through *node_0* relatively inexpensive.

Once the data is distributed, we can begin to parallelize the computations. We intend to have each processor run a copy of the SA on a subgrid, communicating with other processors to resolve data dependencies. To manage this communication and synchronization we designed a fine-grain choreographer to coordinate the processors much in the same fashion as our coarse-grain choreographer coordinates actions between the UI and SA. *Node_0* oversees the parallel action by sending cues the other nodes to initiate communication, synchronization, and computa-

tion. The structure of the parallel simulator is shown in Figure 9.



3.5 Nonlinear Solution

The nonlinear solver creates an initial guess at the solution and then repeatedly asks for a new Jacobian matrix and residual vector to be formed, requests the linear solution of this system, and applies the resulting update until the convergence criteria are met. In the nonlinear solver

module itself, only the initial guess routines require parallel communication. The matrix formation routines and the linear solver reside in separate modules and will be discussed later.

The initial guess routines need non-local information when the creating a new guess at the solution based upon bias conditions. Once again, the boundary conditions cause problems. For example, to compute the extrapolation factor for a projection properly, all bias conditions must be examined. The projection is scaled by the differences between the new bias conditions and the previous values. To obtain this information, it is necessary to have a global picture of the boundary conditions. We evaluated two possible solutions:

- Add a communication phase to the algorithm to make all processors agree upon the correct extrapolation factors. This approach has the disadvantages of adding communication overhead and requiring a moderate amount of code modification.
- Add extra data structures, duplicated on each processor, which contain global boundary condition information. This has the disadvantage of requiring additional data structures to store the duplicated global information. It has the advantages of requiring no communication and minimizing the code changes. (We simply modify the call to the projection algorithm so that the global, rather than the local, boundary conditions are passed.)

We chose the latter solution for three reasons: existing data structures were unaffected, the code modifications were minimal, and the amount of extra data was small and did not impact memory requirements measurably.

3.6 Matrix Form and Assemble

The assembly process proceeds in a finite element fashion, evaluating on a triangle-by-triangle basis. Each triangle forms its local contribution which is then added to the global system. If all information needed during assembly is local to a triangle, assembly may take place independently on all processors. This is usually the case; however, some of the physical models do require non-local data. Unlike in the initial guess routines, there is no simple way to avoid communicating to resolve non-local references as the data needed may be constantly updated by its owner.

Semiconductor devices often contain different materials within a single structure. One example is a silicon-oxide interface (Figure 10) common in MOS structures [9]. The mobility of the carriers along these interfaces differs from the mobility in bulk silicon. If the user desires to model this mobility variation, each silicon triangle along the interface must have knowledge of the material type and present electrical parameters of its neighbors in order to properly compute mobility at the interface. We were forced to modify this model so that interface triangles could

obtain the material parameters of neighboring triangles on remote processors. Before evaluating this mobility model, the fine-grain choreographer initiates a communication phase to exchange the data required for model evaluation. The model itself was modified to access a temporary buffer containing the non-local information. By performing all communication beforehand, code changes were again minimized. Fortunately, models of this type are rare in the current release version of PISCES.

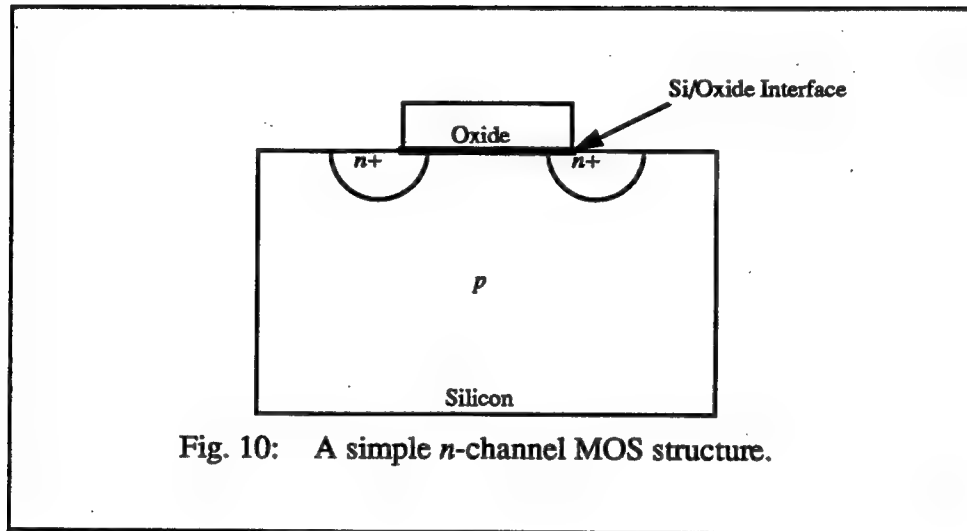


Fig. 10: A simple n -channel MOS structure.

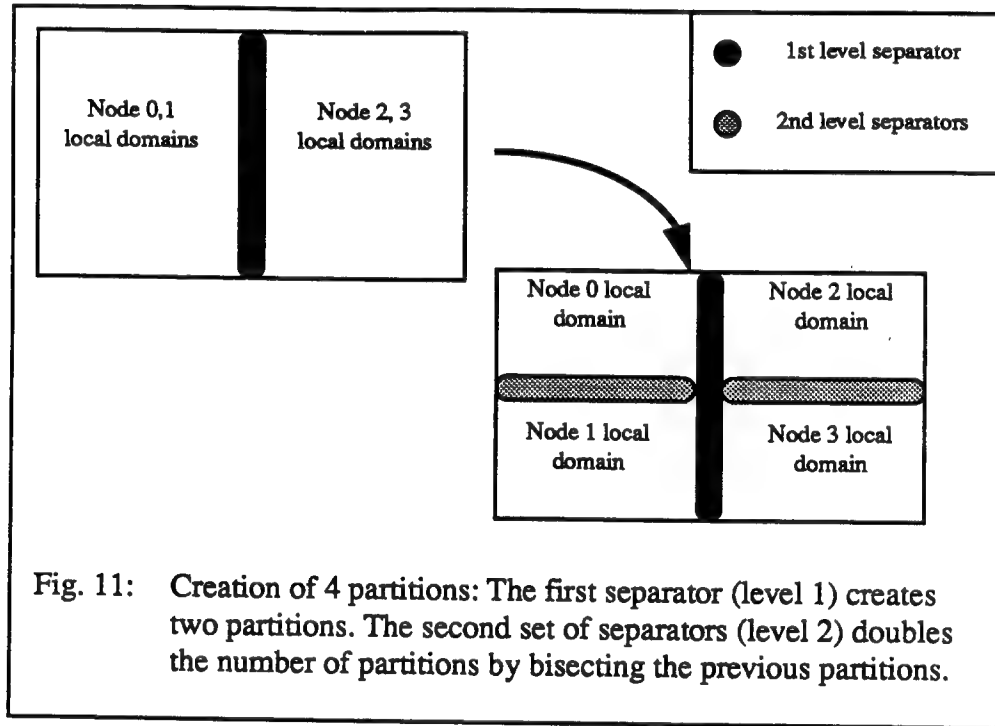
3.7 Linear Solution

The linear solution code is distinct from the rest of the simulator and has a well-defined interface. The linear solver accepts a matrix and a right-hand-side vector from the nonlinear solver and returns a solution vector. As long as the solution is correct and computed efficiently, the linear solver can be treated as a “black box” by the rest of the application. The serial version of PISCES uses a public domain sparse direct solver[7]. Although written more than fifteen years ago, it continues to be quite efficient on most serial architectures. However, adapting this code for parallel execution is problematic as the algorithms and data structures do not map well to modern distributed-memory architectures. Fortunately, the segmentation between the linear solver and the remainder of the code allows us to swap one “black box” for another one with identical behavior. The replacement of the linear solver had no effect on the remainder of the code.

We replaced the existing solver with a distributed multi-frontal (DMF) solver [23, 24] developed in our lab and modelled after the work of [18]. Sparse direct methods generally contain two phases: a symbolic factorization followed by a series of numeric factorizations. The symbolic phase first reorders the equations to minimize non-zero fill-in during elimination. This also mini-

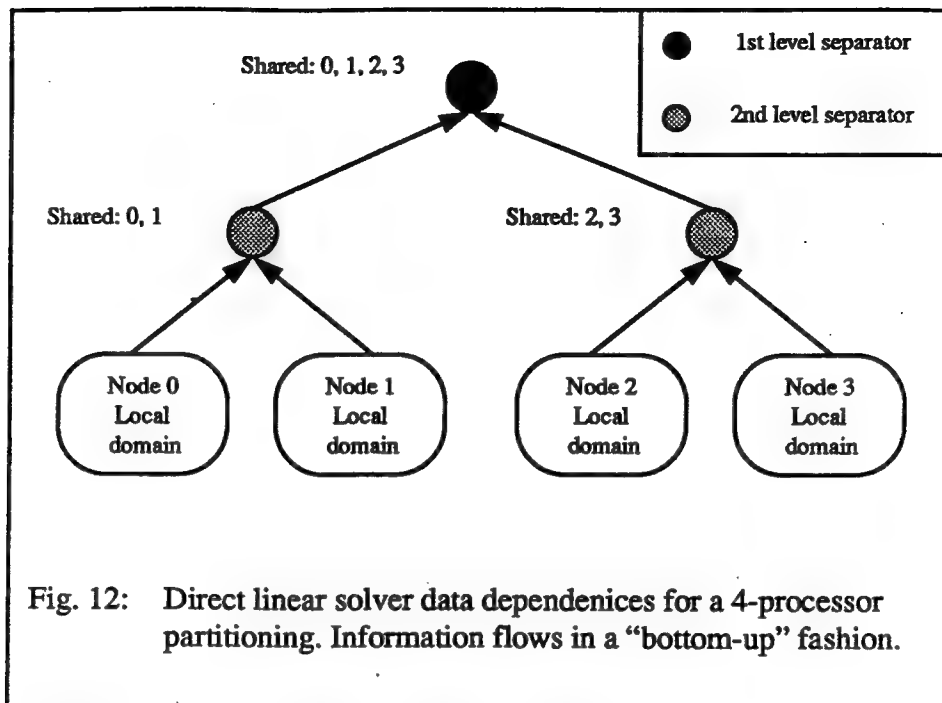
mizes the floating point operations to compute the factor and the memory required to store the factor. Once a reordering has been computed, the matrix is symbolically factored to determine the exact non-zero structure of the factor. Using this information, the requisite data structures are created. The numeric factorization performs the actual floating point work using the elimination ordering and storage prepared by the symbolic phase.

In parallel, the symbolic and numeric factorizations become more complicated due to both the constraints on elimination order posed by the separators and the communication required to factor them. RSB creates partitions in a top-down fashion (Figure 11). We exploit this behavior when computing the order of elimination by factoring the local domains independently followed by the sets of RSB separators. The sets of separators are factored in the inverse order in which they were created by RSB (Figure 12).



To accommodate the constrained order of elimination, a two-phase reordering is used. First, the multiple minimum degree (MMD) algorithm [25] is used to reorder each processor's local domain. Afterward, equations within the separators are added to the elimination ordering. The interiors are factored independently; but, the processors must communicate to factor the separators.

The non-zero fill-in during a sparse direct factorization adds coupling between equations residing on different processors. To insure a correct linear solution, processors must have knowl-



edge of some equations that are resident on other machines. The UI adds the extra grid points that generate these equations to the processors' data. These grid points are used solely by the linear solver and have no effect upon the remainder of PISCES.

3.8 Retrospective: What additional permanent data structures were necessary

In the previous sections we have described the procedures we followed for implementing our parallel PDE solver. To provide a complete picture of the parallelization process, it is useful to describe the major data structures added during development. All of these data structures are used by the choreographers. They are external to PISCES and have no effect upon the existing data structures. They are summarized below:

- *local_to_global_map*[1..#local grid points]: Each processor keeps an array which maps the local node numbering of its subgrid to the global node numbering. All communication among the processors uses the global numbering system.
- *owner*[1..#local grid points]: The array used to determine the processor in charge of accumulating updates to each shared grid point. The owner is responsible for communicating a consistent set of values for the grid point to other processors sharing the node.
- *share_count*[1..#local grid points]: This array, in conjunction with *share_list*, provides a complete picture of the interactions with other processors for each the shared grid point. The data is stored using a quotient list structure [24]. For each grid point, *share_count*

points to the starting location in *share_list* of that node's list of involved processors (Figure 13). The number of processors sharing node *i* is given by *share_count*[*i*+1] - *share_count*[*i*].

- *share_list*[1..*share_count*[#points+1]]: This array (Figure 13) contains lists, one for each grid point, of the processors sharing a grid point. It is indexed by *share_count*.

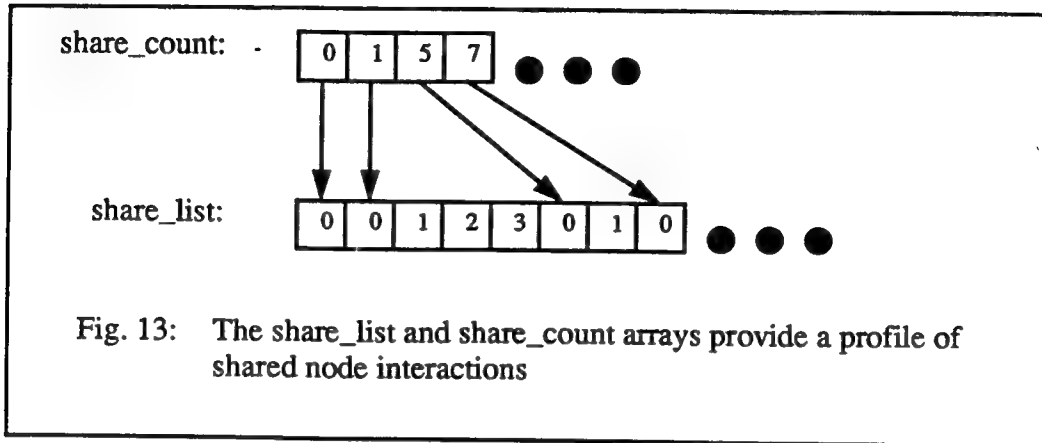


Fig. 13: The *share_list* and *share_count* arrays provide a profile of shared node interactions

- *separator_level*[1..*#local grid points*]: Tells which level separator a shared grid point belongs in. It is used exclusively by the linear solver as described earlier.
- *Global Electrode Data*: This is a complete set of boundary condition data for the global system to insure consistency across the processors as discussed above.
- *elem_owner*[1..*#global triangles*]: This array is used solely by the UI program to determine the proper destination for each triangle. It is not sent to the SA.

3.8 Porting to the CM-5

To prove that our methodology creates applications that can be easily ported to other parallel computers supporting the SPMD model, we moved the code to the CM-5. Although the CM-5 was originally designed to support the dataparallel programming model[26], it also supports the SPMD programming model through its CMMD message passing library[27]. Each processing node consists of a SPARC processor controlling four floating-point vector units. Unfortunately, the vector units can only be utilized by programs written using a dataparallel language[28]. As declared earlier, we were unwilling to make wholesale changes to the code. This meant that we were confined to the SPARC processor and unable to access the high-performance vector units from our application. However, the machine still offers reasonable performance as demonstrated in our results section.

The porting process was surprisingly straightforward. Most of the work involved changing

the UI/SA communication layer. Since the CM-5 has one (or more) SUN workstation equivalents as control processors, we chose to run the UI on a control processor rather than a remote workstation. The control processors provided acceptable performance and provided a 20Mb/s connection connected to the attached parallel machine, a considerable improvement over TCP/IP communication. Also, the host can communicate directly with all nodes, making the intermediate hop through *node_0* unnecessary. The host uses the same byte ordering as the parallel processors, making byte reordering unnecessary.

Otherwise, we translated NX calls into corresponding CMMD calls. In most cases, a simple wrapper was sufficient and the existing code was left untouched. In rare cases, such as the hypercube specific *cubedim()* function, we substituted functions more appropriate to the CM-5. The porting work required less than two weeks and minimal code changes were necessary.

4. Results

Using the new parallel device simulator, we can scale the computational resources to match problem requirements. This should provide not only the ability to simulate existing grids in less time but also the ability to simulate large, complex grids which are computationally infeasible on serial architectures. In order to demonstrate the utility of our new parallel application, we ran the simulation of the CMOS inverter structure described in Section 2 (a total of 29 bias steps) on several different machines. We simulated four varying grid sizes as described in Table 1. They

Table 1: Simulation grid statistics

	2K Grid	8K Grid	11K Grid	18K grid
Number of grid points	1,607	7,844	10,728	18,503
Number of triangles	2,963	15,158	20,873	36,400
Number of equations	4,821	23,532	32,184	55,509

comprise a realistic set of grids to study various characteristics of this complex device. The lower resolutions are suitable to model the macroscopic behavior. The finer resolutions are required to capture small-scale effects within the device.

Only the CM-5 had sufficient memory to simulate the 18k grid. The iPSC/860 contains enough total memory to run the 18K grid on 16 or 32 processors; however, a poor load balance caused a small subset of the processors to run out of memory. In the following tables, an "N.A."

entry indicates that insufficient memory existed to run the simulation.

4.1 Serial Timings

To provide a set of serial benchmarks for comparison, we ran the simulation on the three fastest serial computers available in our lab. The simulation times are shown in Table 2. Not sur-

Table 2: Serial Solution Times For CMOS Inverter

	2K Grid	8K Grid	11K Grid
SUN 4/670	1,214s	40,032s	N.A.
IBM RS/6000 Model 530H	417s	11,263s	23,822s
IBM RS/6000 Model 560F	279s	7,239s	15,687s

prisingly, the IBM Model 560F, the fastest floating point processor, completed the simulations in the shortest time. All of the comparisons between the serial simulator and the parallel simulator are based upon the corresponding time on the Model 560F. The other solution times are included for reference.

4.2 Intel iPSC Timings

We ran the parallel code using the SUN 4/670 as the front end for our 32-processor Intel iPSC/860. For the small number of bias points in this simulation, the setup time using TCP/IP contributes greatly to the runtime. For example, the data transmission to set up a 32-node partition takes roughly fifteen minutes. Had we run a more thorough simulation, the setup time would have been better amortized. The timings shown in Table 3 include all setup time. To provide a better understanding of the asymptotic behavior of the parallel code, Table 4 contains timings with the data transmission times subtracted. In Figure 14, we have plotted the best serial solution time for each grid size as well as the best iPSC solution times both including and discounting the setup communication overhead.

Table 3: iPSC Solution Times for CMOS Inverter

	2K Grid	8K Grid	11K Grid
iPSC/860 1PE	1,515s	N.A.	N.A.
iPSC/860 2PE	829s	N.A.	N.A.

Table 3: iPSC Solution Times for CMOS Inverter

	2K Grid	8K Grid	11K Grid
iPSC/860 4PE	635s	N.A.	N.A.
iPSC/860 8PE	597s	2,687s	3,611s
iPSC/860 16PE	799s	1,900s	3,231s
iPSC/860 32PE	1279s	2,298s	3,107s

Table 4: iPSC Solution Times for CMOS Inverter without TCP/IP

	2K Grid	8K Grid	11K Grid
iPSC/860 1PE	1,480s	N.A.	N.A.
iPSC/860 2PE	778s	N.A.	N.A.
iPSC/860 4PE	522s	N.A.	N.A.
iPSC/860 8PE	372s	2,462s	3,386s
iPSC/860 16PE	349s	1,450s	2,781s
iPSC/860 32PE	375s	1,394s	2,203s

For the 2K grid, the iPSC cannot outperform the IBM 560F. This first observation begs the question: why did we develop a parallel simulator? The answer is: we need the ability to solve existing large problems quickly and the ability to solve larger problems which are computationally infeasible on current serial machines. For small problems, the overhead of setup, communication, and synchronization may make some simulations run slower in parallel than serially. The 2K grid is such a problem. However, the 8K grid is more encouraging. We achieve 4x speedup over the IBM including overhead and greater than 5x speedup with overhead discounted. The 11K grid is more encouraging still. We obtain a 5x speedup with overhead and a 7x speedup without it.

Figure 14 shows that as grid sizes increase, the solution times of the parallel code increase much more slowly than the serial solution times. The ability to scale more efficiently with grid

size will prove significant as the sizes and runtimes of simulations grow.

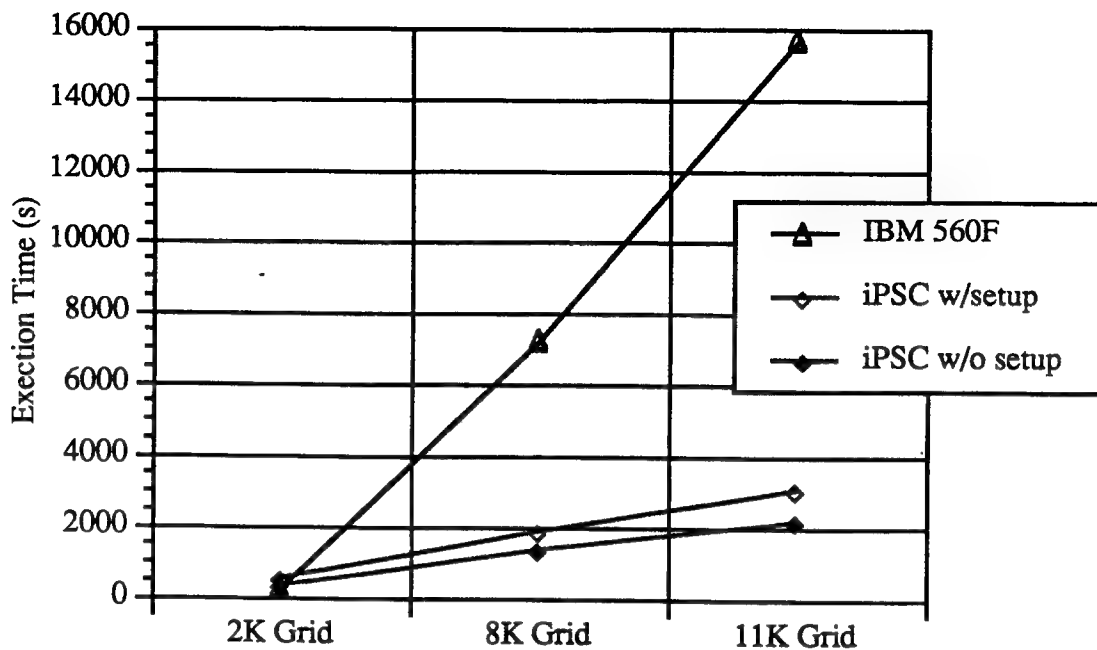


Fig. 14: Graph of the best solution times for the IBM 560F and the iPSC/860 .

4.3 CM-5 Timings

We ran the simulations on two CM-5e's. (The CM-5e uses the newer SUN Viking processor in place of the original SPARC processors.) The CM-5 manages to barely outrace the IBM

Table 5: CM-5 Solution Times for CMOS Inverter

	2K Grid	8K Grid	11K Grid	18K Grid
CM-5e 32PE	262s	1,150s	1,792s	3,054s
CM-5e 64PE	270s	832s	1,232s	2,094s

560F on the 2K grid. For the 8K grid and the 11K grid, we observe speedups over the IBM of nearly 9x and 13x, respectively. Finally, with the 18K grid, we demonstrate the ability to simulate problems computationally infeasible on our serial machines. The ability to scale computational resources makes this simulation possible. Plotting the data (Figure 15) reinforces our assertion

that the parallel simulator offers superior scalability as problem sizes increase.

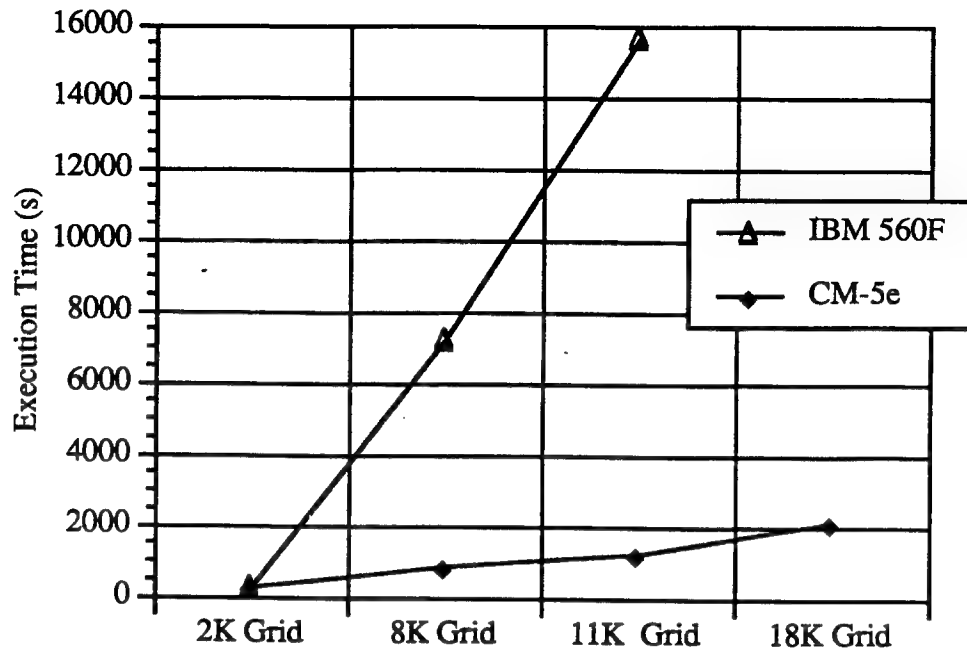


Fig. 15: Graph of the best solution times for the IBM 560F and the CM-5e.

5. Conclusions

In this paper, we have demonstrated that PISCES, a commercially-supported “dusty-deck” PDE solver, can be adapted for parallel execution with minimal changes to the existing serial code. In this fashion, we have retained the value captured in the long-term development of the program. The proven portability of our methodology allows the code to easily migrate to other distributed-memory architectures. We believe the methodology described in this paper should work well for any PDE solver with a similar program structure.

Maintaining the original code and data structures does not prevent the parallel implementation from providing striking reductions in simulation times for realistic large grids. We have already demonstrated order of magnitude (or more) improvements in simulation times for current problems as well as the ability to simulate grids too large for our serial computers. In fact, the behavior of the parallel code as grid sizes increase suggests a vast potential for simulating large and ultra-large structures. In the semiconductor industry, the competitive advantage gained could have immeasurable benefits.

Acknowledgments

We would like to thank Horst Simon for providing the spectral partitioning code. We also like to thank Lennart Johnsson and Thinking Machines Corporation for providing access to their computing resources. This research was sponsored by the State of California's Department of Commerce under grant number C90-072 and by DARPA.

References

- [1] M.T. Heath, "The hypercube: a tutorial overview," In *Hypercube Multiprocessors*, pp.7-10, SIAM, Philadelphia, PA, 1986.
- [2] M.R. Pinto, C.S Rafferty, and R.W. Dutton, "PISCES-II: Poisson and continuity equation solver," Stanford Electronics Lab, Stanford Univ., Stanford, CA, 1985.
- [3] M.R. Pinto, C.S Rafferty, H.R. Yeager, and R.W. Dutton, "PISCES-IIB Supplementary Report," Stanford Electronics Lab, Stanford Univ., Stanford, CA, 1985.
- [4] B.V. Ghokale, "Numerical solutions for one-dimensional silicon n-p-n transistor," *IEEE Trans. Electron Devices*, Vol. ED-17 (1970), pp. 594-602.
- [5] H.K. Gummel, "A self-consistent iterative scheme for one-dimensional steady state transistor calculations," *IEEE Trans. Electron Devices*, Vol. ED-11 (1964), pp. 455-465.
- [6] C.S. Rafferty, M.R. Pinto, and R.W. Dutton, "Iterative methods in semiconductor device simulation," *IEEE Trans. Electron Devices*, Vol. ED-32 (1985), pp. 2018-2027.
- [7] D.A. Calahan and P.G. Buning, "Vectorized General Sparsity Algorithms with Backing Store," SEL Report 96, Systems Engineering Laboratory, University of Michigan, Ann Arbor, MI, 1977.
- [8] Z. Johan, "Data parallel finite element techniques for large-scale computational fluid dynamics," Ph.D. Thesis, Stanford University, 1992.
- [9] B. Nour-Omid, A. Raefsky, and G. Lyzenga, "Solving finite element equations on concurrent computers," *Procedures of the Symposium on Parallel Computations and their Impact on Mechanics*, Boston, 1987, ASME, 1988.
- [10] K. Wu, R.F. Lucas, Z. Wang, and R.W. Dutton, "New approaches in a 3-D one-carrier device solver," *IEEE Trans. CAD*, Vol. 8, No. 5 (1989), pp. 528-537.
- [11] V. Venkatakrishnan, H.D. Simon and T.J. Barth, "A MIMD implementation of a parallel Euler solver for unstructured grids," *The Journal of Supercomputing*, 6 (1992) 117-127.
- [12] P.J Hatcher and M.J. Quinn, *Data-parallel Programming on MIMD Computers*, MIT Press, Cambridge, MA, 1991.
- [13] H.D Simon, "Partitioning of unstructured problems for parallel processing," *Computing Systems in Engineering*, 2 (1991) 135-148.
- [14] A. Pothen, H.D. Simon, and K.P. Liou, "Partitioning Sparse Matrices with Eigenvectors of Graphs," *SIAM J. Mat. Anal. Appl.*, 11 (1990), pp. 430-452
- [15] J.G. Malone, "Automated mesh decomposition and concurrent finite element analysis for hypercube computers", *Comp. Meth. Appl. Mech. Eng.*, Vol. 70, No., 1, (1988), pp. 27-58.
- [16] C. Farhat and M. Lesoinne, "Automatic partitioning of unstructured meshes for the parallel solution of problems in computational mechanics," *Internat. J. Numer. Meth. Eng.*, Vol. 36, No. 5, (1993), pp. 745-764.
- [17] C. Farhat, "A simple and efficient automatic FEM domain decomposer," *Computers and*

- Structures, Vol. 28, No. 5 (1988), pp. 579-602.
- [18] R.F. Lucas, "Solving Planar Systems of Equations on Distributed-Memory Multiprocessors," Ph.D. Dissertation, Stanford Univ., 1987.
 - [19] J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Mateo, CA, 1990.
 - [20] iPSC CIO Ethernet Reference Manual, Intel Scientific Computers, Beaverton, OR, 1990.
 - [21] iPSC Concurrent Programming Reference Manual, Intel Scientific Computers, Beaverton, OR, 1990.
 - [22] R.S. Muller and T.I. Kamins, *Device Electronics for Integrated Circuits*, John Wiley and Sons, New York, 1977.
 - [23] I.S. Duff, "Parallel implementation of multifrontal schemes," *Parallel Computing*, Vol. 3, pp. 193-204.
 - [24] I.S. Duff, A.M. Erisman, and J.K. Reid, *Direct Methods for Sparse Matrices*, Oxford University Press, London, 1986.
 - [25] A. George and J.W.H. Lui, "The evolution of the minimum degree algorithm," *SIAM Review*, Vol. 31, pp. 1-19, 1989.
 - [26] *The Connection Machine CM-5 Technical Summary*, Thinking Machines Corporation, Cambridge, MA, 1992.
 - [27] *CMMD Reference Manual, Version 3.0*, Thinking Machines Corporation, Cambridge, MA, 1993.
 - [28] *CM Fortran Reference Manual, Version 2.0 Beta*, Thinking Machines Corporation, Cambridge, MA, 1992.

Layout-based Extraction of IC Electrical Behavior Models

K. Wang, F. Rotella, T. Chen, D. Yang, A. Lee,
Z. Yu, R.W. Knepper*, J. Watt⁺ and R.W. Dutton

Stanford University
Center for Integrated Systems
Stanford, CA 94305

* CIS Visitor from IBM, East Fishkill, NY
⁺ Cypress Semiconductor, San Jose, CA

Abstract

Behavior of IC structures is modeled using a heterogeneous set of tools and derived physical representations. A unified 3D information model is demonstrated with special emphasis on application of solid geometry modeling techniques. Examples used in this presentation include modeling of SRAM technology and interconnect structures that include packaging considerations as well. Issues of mixed level simulations are considered based on circuit and thermal constraints on IC structures.

Introduction

There are a growing number of technology and circuit design problems where the interactions between layout constraints and the underlying technology base create situations that require detailed TCAD analysis. This demonstration shows progress in supporting mask-to-behavior modeling. It involves the integration and demonstration of standard interfaces and tools--GDS2, ACIS (solid modeling) [1], PISCES and SPICE--as well as evolving standards and prototype tools such as SWR (Semiconductor Wafer Representation) [2] and VIP3D, Stanford's "3D Virtual Process" application. The focus of the demonstration will be on extracting IC cell behavior, primarily using two SRAM examples, based on layout input and tools that fully exploit the 3D nature of the problem. The goals of this work are two-fold: 1) to extend simulation capabilities in support of fully 3D considerations and 2) to revolutionize tool inter-operability through the use of a consistent information model--a major challenge facing both users and developers of TCAD. Traditionally, simulation of IC processes and circuit designs have been done using one- and two-dimensional models. The integration of simulators, where devices and circuits are connected, has been especially difficult due to the lack of common representations and means to extract the needed subsets of the data. For example, circuit extractions rely on 2D layout information (i.e. X-Y plane) whereas device simulators deal with 2D cross-sections (i.e. Y-

Z plane). By means of the fully 3D geometry and gridding tools used in this work, along with the utilities that derive consistent subsets of the data, both classes of simulation tools can be supported and extended into the 3D domain. Moreover, through use of 3D solid modeling capabilities, a central and universally accessible information model is developed which serves to integrate tools and assure greater data consistency.

Summary of Results

The following discussion and set of examples give a summary of typical demonstration results to be exhibited. The intent is to walk the potential user through a design process going from layout through various stages of behavioral modeling. Fig. 1 (see final page) shows an architectural overview of information flow through this design process.

A. Extraction of Geometry from IC Layout

Simulation flow begins with the construction of devices based in part on mask information from an IC layout tool. The Cadence tool is used in creating the data and passed through a standardized interface with GDS2. In Fig. 2, a 6 transistor SRAM cell has been created with VIP3D, a tool which uses this mask information in conjunction with technology param-

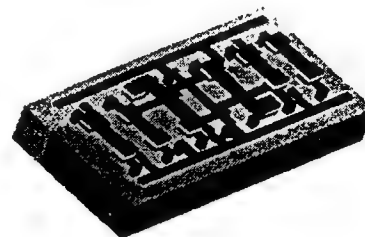


Fig. 2: Model of six transistor SRAM, created using VIP3D based on solid geometry manipulations. Cut-plane (2D) is used as part of mixed-mode analysis (see Fig. 5).

ters such as layer thicknesses as well as 2D simulation results to build up 3D geometries describing the structures created on a wafer. The SWR interface to solid-modeling is used by applications such as VIP3D to create, modify and maintain this wafer state data. In this way, VIP3D performs wafer processing and provides the user with a way to describe devices in 3D.

A more complex SRAM geometry based on a commercial cell design is shown in Fig. 3a, which was produced in collaboration with Cypress Semiconductor using the same basic methodology and solid modeling. Creation of this cell involved sweeping of arbitrary cross-sections along mask edges and resolution of step coverage during conformal deposition. The layer thicknesses and the bird's beak profile for the field oxide are based on either process simulation or measurement. Fig. 3b shows a 4X4 array assembled from these cells indicating the more complete configuration of second layer metal which will affect interconnect capacitances.

B. Deriving 2D Cross-Sections from Solid Geometry

Based on such 3D device models, both 2D and 3D device simulation can be run. One major advantage for having a single 3D device information model is the consistency of the data model, which ensures that simulation tasks can be performed in a unified manner. With the specification of a 2D cut plane,

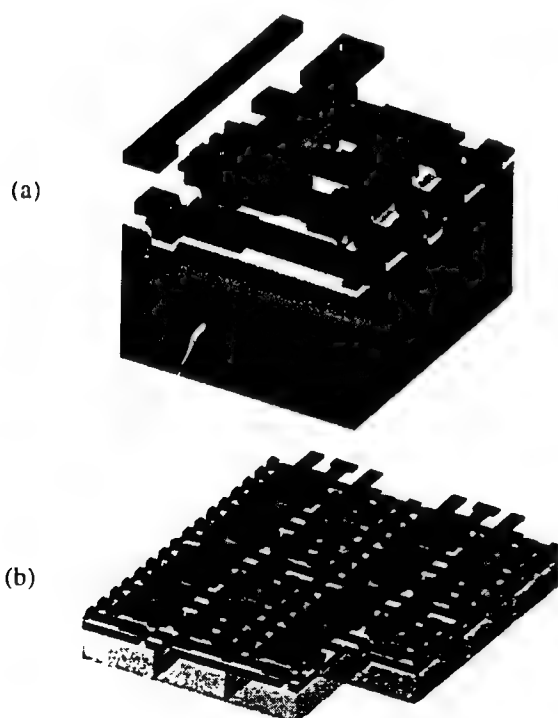


Fig. 3: Commercial SRAM modeled using solid geometry approach: (a) 6T cell and (b) array of cells.

such as the one shown in Fig. 2, the system can extract device geometry information through attributes which include various physical properties such as: material type, analytical or simulated doping distributions. In this way, 2D subsets of the complete 3D model can be obtained.

C. Gridding of 2D Surfaces and 3D Volumes for Simulation

In addition to geometrical cross-sectioning, various mesh services are also provided, including 2D cross section mesh, 2D surface mesh and 3D volume mesh. The mesh generation engines are based on quadtree(2D) and octree(3D) techniques with two major improvements: level control functions for controlling the mesh density and delta-zone (and warping) functions for optimizing the mesh quality along non-planar boundaries. To generate 2D surface mesh, the system first extracts all external boundary faces and then performs 2D mesh generation on each of these faces.

D. Lumped Parameter Extraction of Interconnect Parameters

One area in which 2D surface meshing of 3D volumes can be particularly useful is in parasitic analysis of devices and interconnect structures, which is a very important step in the design verification process. Traditional IC parasitic extraction is based only on the 2D layout artwork and is not well-suited to capture true parasitic behavior of devices when non-ideal planarizations and process variations are considered. For interconnect modeling, 2D electrical field analysis is proving insufficient in accounting for increasing fringe effects caused by shrinking interconnect size and closer line pitch. In Fig. 4 we see a surface mesh of the 3D interconnect structure of the SRAM cell in Fig. 2. By using this surface-meshed structure and the 3D capacitance extraction application FASTCAP from MIT [3], parasitic extraction for 3D models can be achieved.

E. Mixed-Mode Simulation for Extraction of Electrical Behavior

Cross sections and volumes of VIP3D results can also be used in mixed-mode simulations to obtain circuit performance as

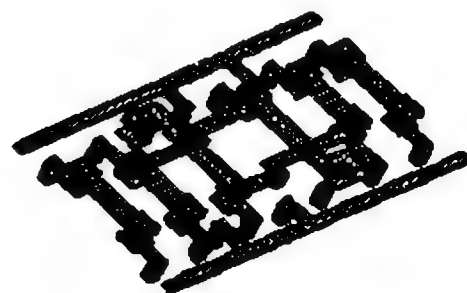


Fig. 4: Surface meshing of poly and metal layers for SRAM (see Fig. 1).

shown in Fig. 1. In this work a generalized interface between the Berkeley SPICE tool and numerical device simulators such as Stanford's PISCES (or IBM's FIELDAY) are demonstrated. This interface allows a designer to examine the circuit performance of a new device for which a SPICE analytic model does not exist or may be insufficient. Such inadequate models include those for short channel MOSFET's, GaAs MESFET's, and optoelectronic LED structures. In addition, more complex effects such as self heating and photon generation can be simulated by the numerical device and hence, examined in the circuit operation. The emphasis in this particular demonstration will be on illustrating the ability to quickly select cross-sections and merge device and circuit levels of abstraction into a unified simulation environment.

F. Examples using SRAM Technologies

In order to illustrate the utility of the overall system, the SRAM of Fig. 2 is used as a cell-based design where performance is of major interest to the designer. In this example, two aspects of the cell are examined. Fig. 5a shows the circuit diagram with the basic cell and control circuitry. The shaded

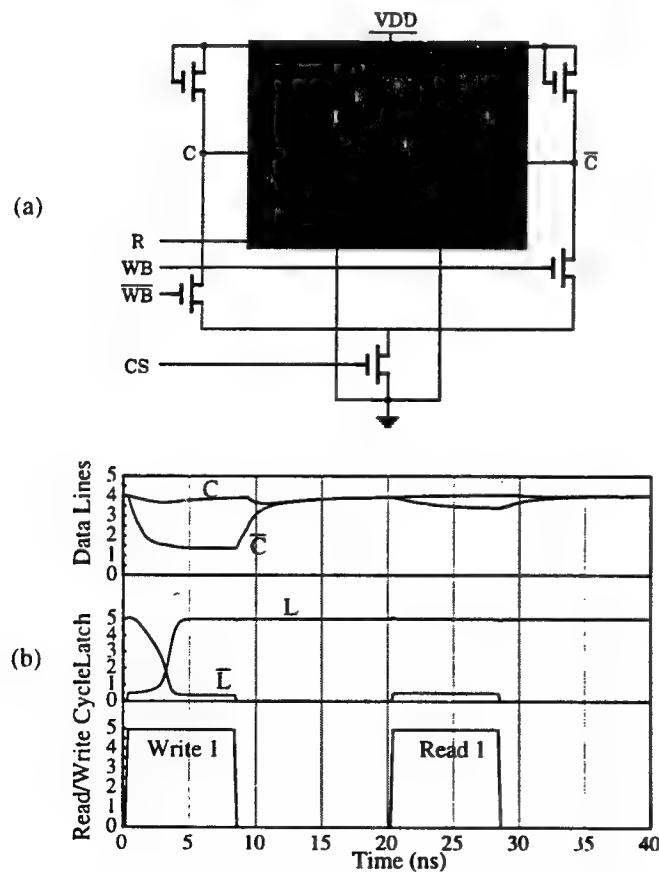


Fig. 5: Mixed-mode analysis of SRAM cell: (a) circuit schematic where shaded region represents device level modeling; (b) output waveforms based on coupled circuit- (SPICE) and device- (PISCES) level analysis.

region is simulated with PISCES numerical devices while the other control transistors are simulated with analytic circuit models. In addition, parasitics may be included for the line interconnects, as calculated with FASTCAP. Fig. 5b shows the transient behavior of the circuit for a write/read cycle. The major concern is how quickly C and C (bar) come back together once the pass transistors are turned off.

The more complex SRAM of Fig. 3 shows a cell where the tight design rules at the cell level can produce both manufacturability and circuit performance challenges. Presently the examples to be demonstrated for this commercial SRAM focus primarily on the geometric specification and parameterization issues rather than on the simulations. Nonetheless, by utilizing mixed-mode capabilities demonstrated above, the design can be examined for potential problems at both levels which can then be directly linked to the layout data. In addition, perturbations on the layout can be used to fix layout or performance related problems that may become uncovered.

Conclusion

An integrated system for 3D device and circuit characterization based on layout and process information is presented. Use of a geometrical wafer representation provides a consistent and effective means of manipulating 3D IC models. By means of sectioning and gridding tools, flexibility in moving from 3D wafer-level analysis to 2D and 3D interconnect-, device- and circuit-level simulation has been demonstrated. The supporting services allow designers to characterize circuit behavior of IC's directly from a layout tool. All of these capabilities will be demonstrated primarily using the two SRAM cell examples discussed above.

Acknowledgments

The authors gratefully acknowledge research support from ARPA (contracts #DAAL 03-91-C-0043 and #DABT 63-93-C-0053) and SRC (contract #94-YC-7074). Collaborations with Cypress Semiconductor have been most helpful and encouragements from Dr. Tony Alvarez are very much appreciated. Initial contributions by Eric Tse and Melissa Cheek dealing with 3D information modeling (et3D) and 2D mask information have also been very valuable.

References

- [1] ACIS, Spatial Technology, Inc., Boulder, CO, 1992
- [2] TCAD Framework Group, Semiconductor Wafer Representation Working Group, "Semiconductor Wafer Representation Procedural Interface, Version 1.5," CAD Framework Initiative, Inc., Austin, TX, September, 1993.
- [3] K. Nabors and J. White, "A Fast Multipole Algorithm for Capacitance Extraction of Complex 3-D Geometries," *Proc. Custom Int. Circuits Conference*, San Diego, CA, 1989, pp. 21.7.1 - 21.7.4.

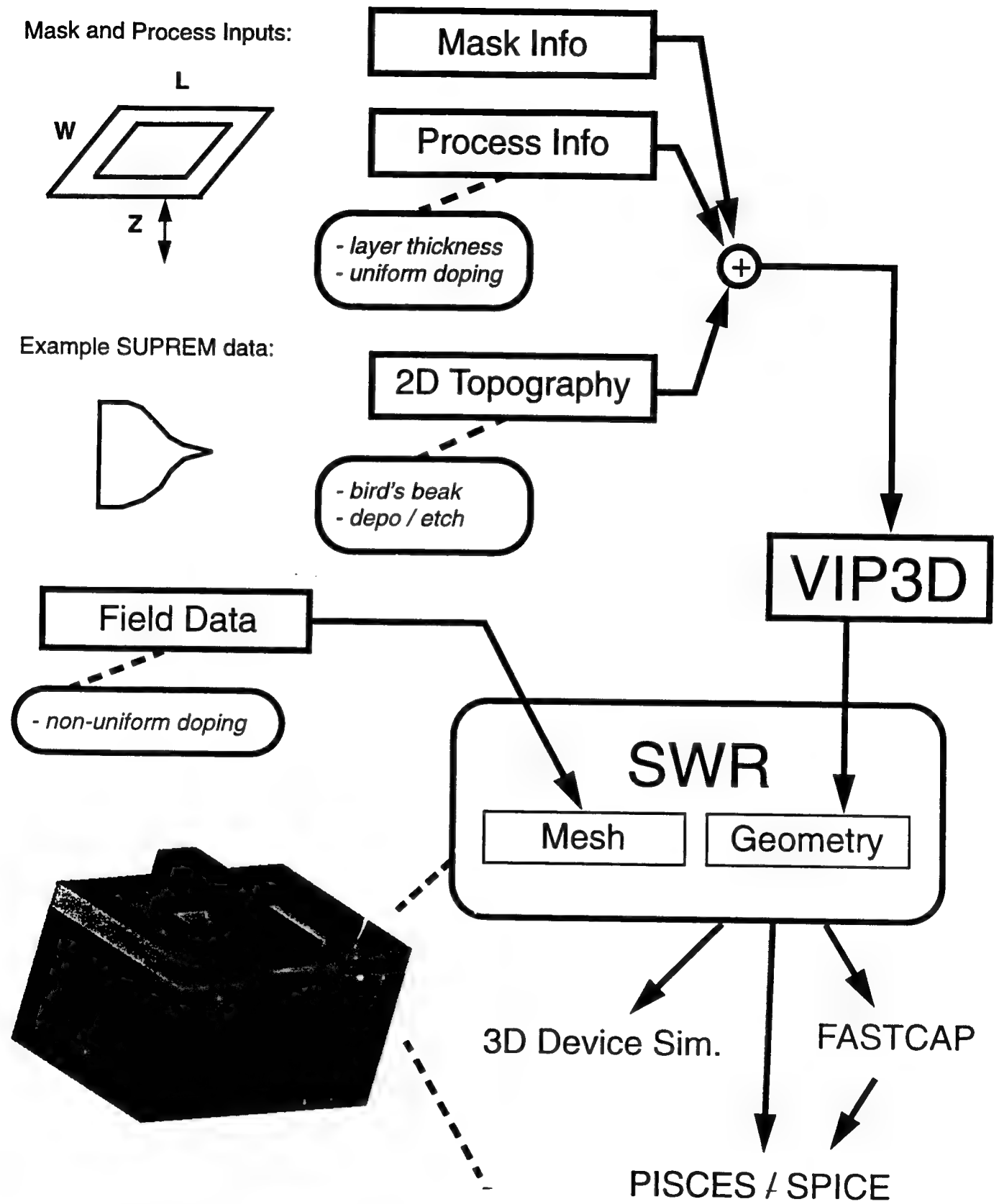


Fig. 1: Schematic model and associated information flow used in both VIP3D and subsequent device, interconnect and circuit analysis.

Grid Evolution for Oxidation Simulation Using a Quadtree Based Grid Generator

Zakir H. Sahul, Eugene W. McKenna, Robert W. Dutton
AEL 231, Stanford University, Stanford, CA 94305-4055, USA

Abstract

Grid and geometry movement algorithms for oxidation simulation are presented using the grid/geometry server *Forest*. Initial grid generation is performed using a quadtree region decimation algorithm. An oxidation solver is used to compute node velocities and the maximum time step is determined from geometry and grid considerations. The boundary nodes are then moved and geometry singularities like loop formation and region crashes are detected and removed. All the mesh nodes are moved and retriangulation is performed if necessary. The quadtree is rehashed and new triangles are allocated to conform to the new geometry. This algorithm guarantees a high grid quality at all times with minimal grid changes between time steps.

1 Introduction

Grid operations for oxidation simulation present many difficulties due to moving boundaries. In addition to grid quality and adaptation requirements, oxidation simulation requires controlled grid movement. Triangles should not diminish to zero areas, nodes should not overtake one another, and the geometry after grid movement should be valid [1] [2]. Boundaries should not self-loop and isolated regions should not move into one another. A grid movement algorithm that addresses all these issues has been developed and incorporated into *Forest*, a 2D geometry and grid server. The flowchart for the entire algorithm is shown in Figure 1.

Grid generation in *Forest* is performed using a quadtree procedure [3] by enclosing the geometry in a root square quadrilateral and recursively dividing it into a terminated lines square mesh. Boundary quadrilaterals are preprocessed by a technique called warping and final triangulation is performed using templates. Note that the geometry, terminated lines mesh and triangular mesh are stored and manipulated independently of one another for simplified grid and geometry operations.

2 Grid Movement

Node velocities are computed using SUPREM-IV's oxidation solver [2] on the triangular mesh. After an appropriate time step is determined, all the nodes in the triangular mesh are moved and a new geometry is computed. The triangular mesh is regenerated if elements

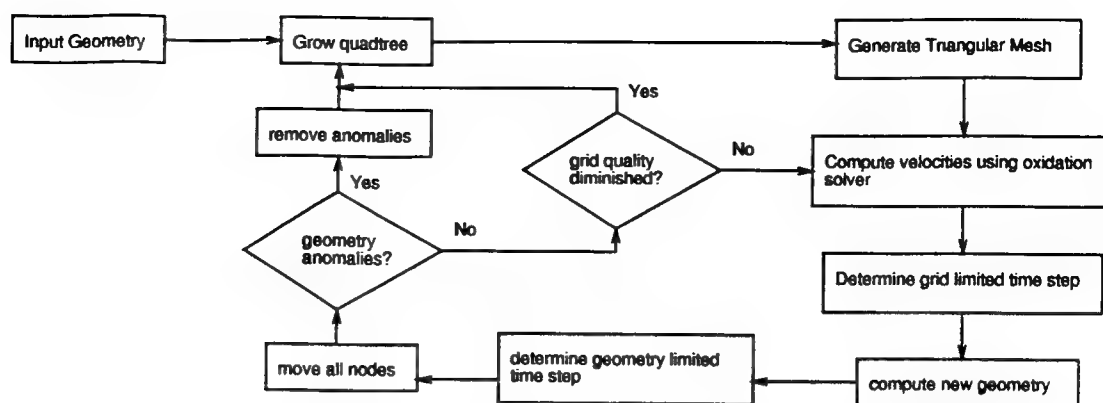


Figure 1: Flowchart for grid evolution during oxidation simulation

become highly deformed. Grid regeneration proceeds by unwarping the quadrilaterals and rehashing the quadtree to reflect the new geometry. If the geometry extends outside of the old root quadrilateral, a new root quadrilateral is formed by recursively doubling the old root's size. All of the branches in the quadtree are then visited and pruned or grown. If a branch that previously enclosed an area of the geometry now does not enclose a material area, it is pruned. Conversely, if an old branch encloses an area when it did not previously, it is grown. The new quadrilaterals are then warped and triangulated. This procedure is illustrated in Figure 2. This strategy guarantees a high quality grid at all times.

Solutions on the new nodes are interpolated from the old triangle mesh and the old triangular mesh is destroyed. The advantage of this approach is that the quadtree structure changes *only* along the moving oxide interface. The triangulation nodes in areas of no grid movement are the same - resulting in fewer interpolations between meshes. Moreover, the level of recursive division of the quadtree can be controlled to make boundary quadrilaterals or quadrilaterals in areas of high solution variation finer, allowing simultaneous grid movement and adaptation.

The maximum time step is determined using the grid and geometry criteria illustrated in Figure 3. In each case, the nodal velocity vectors are shown along with node movement with too large a time step and with the optimal time step. Node overtake conditions are detected by checking for negative or zero triangle areas after the entire triangular mesh is moved.

3 Geometric Singularities

The geometry functions of *Forest* are used to detect and remove region crashing and boundary self-looping conditions. The structure geometry is independent of the quadtree and triangular meshes and consists of boundaries with ordered and directed edges. Nodes along the boundary are first moved to determine a new geometry. Geometry anomalies are then detected. As shown in Figure 3, region crashing is detected by checking for intersections of edges of different region boundaries. Boundary self-looping is detected by checking for intersections between non-adjacent edges. The time step is adjusted using a time inter-

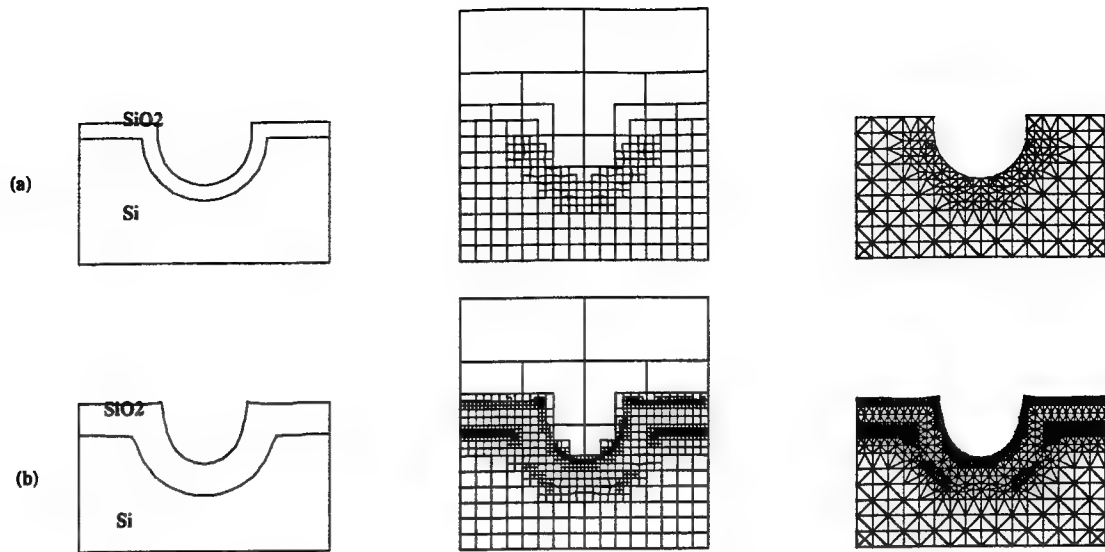


Figure 2: Use of Forest in oxidation simulation. Geometry, quadtree mesh and triangular meshes before (a) and after several oxidation steps (b).

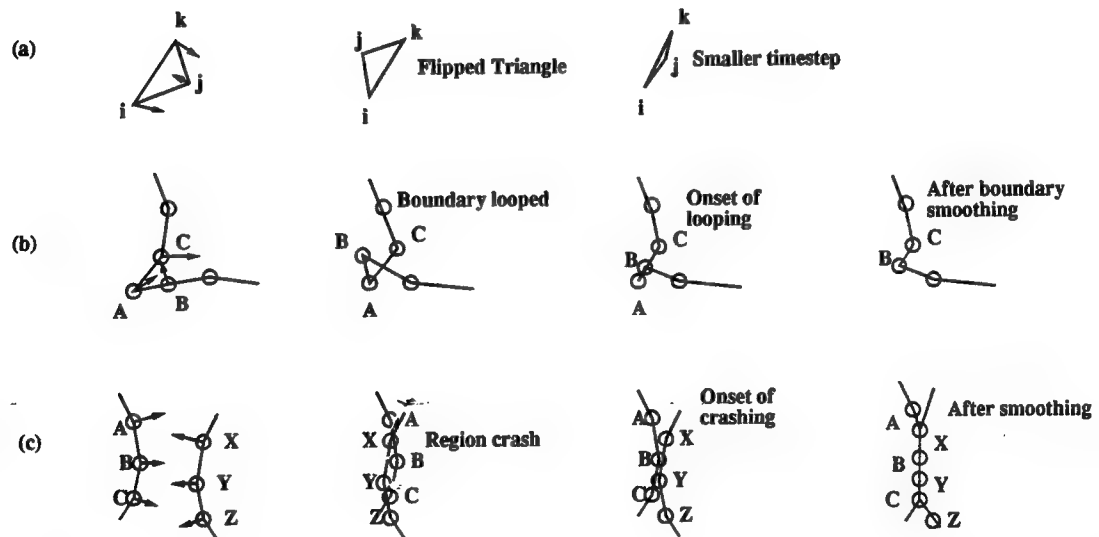


Figure 3: Grid and geometry criteria used to limit time step. (a) Nodes should not overtake one another. (b) Boundaries should not form self loops. (c) Regions should not crash into one another.

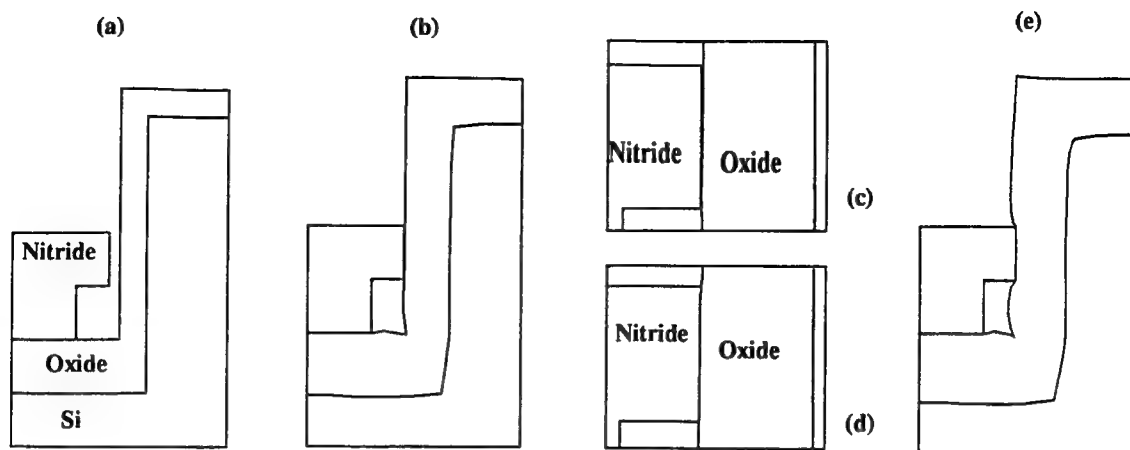


Figure 4: Oxidation simulation with colliding regions. (a) Initial geometry. (b) Geometry at the onset of crashing during oxidation. (c) Close-up view of regions before crashing. (d) Boundary smoothing to allow proper geometric coupling. (e) Subsequent oxidation

val bisection approach until the onset of an anomaly as shown in Figure 3. User-defined closeness threshold criteria are used to determine the onset of anomalies.

To continue the simulation after the onset of region crashing or boundary self-looping, the boundary is smoothed by collapsing nearby edges as illustrated in Figure 3. This removes geometry anomalies and provides proper coupling between different regions. The quadtree grid is updated to conform to the new geometry as discussed above.

Figure 4 illustrates the utility of this approach in oxidizing complicated structures. The sidewall oxide crashes into the abutting nitride region. Time steps are chosen until the onset of the crash and the geometry is smoothed as shown in Figures 4c and 4d. The grid is updated and simulation continued (Figure 4e).

Acknowledgement: This work was supported by the Semiconductor Research Corporation under contract # 92-SP-101.

References

- [1] P. Sutardja. "Finite Element Methods for Process Simulation Application To Silicon Oxidation." Ph.D Thesis. Dept. of Elect. Eng. UC-Berkeley. 1988.
- [2] C. Rafferty. "Stress Effects in Silicon Oxidation - Simulation and Experiments." Ph.D. Thesis. Dept. of Elect. Eng. Stanford University. 1989.
- [3] Z. Sahul, R. Dutton, M. Noell. "Grid and Geometry Techniques for Multi-Layer Process Simulation". Proc. SISDEP '93. Sept. 1993.

Grid Techniques for Multi-Layer Device and Process Simulation

Zakir H. Sahul, Eugene W. McKenna, Robert W. Dutton
Integrated Circuits Laboratory, Stanford University
Stanford, CA 94305

Abstract

A 2D numerical grid generation program suitable for device and process simulation is presented. The program meets the unique grid requirements encountered in multi-layer process simulation - rapidly changing topographies, and use of mesh based and string based TCAD tools.

Grids can be generated for the PISCES-II and SUPREM-IV device and process simulation programs. Adaptive gridding has been achieved for SUPREM-IV diffusion simulation and a prototype interface has been demonstrated for the etching/deposition simulator SPEEDIE.

Introduction

A grid module suitable for multi-layer device and process simulation must be capable of gridding complex geometries, adapting the grid based on various criteria, and being responsive to rapid topography changes. It must also be suitable for use with a wide variety of TCAD tools each with different internal data representations. *Forest*, a 2D automatic quadtree based grid program, provides grid functions with all these capabilities.

Grid Generation

Forest stores the device geometry as an hierarchy of points, edges, boundaries and regions. Each region can contain a number of boundaries allowing structures with voids. String based TCAD tools use and update this geometry data without directly manipulating the grid.

Grid generation (Figure 1) is performed by a quadtree decimation technique (2). The device geometry is surrounded with a root square which is recursively decimated resulting in a final terminated lines quadrilateral mesh. Neither the root square nor the resulting quadtree depends on the specific vertices of the geometry; they only depend on the spatial extent of the structure. The level of decimation can be controlled by a number of factors including doping variation, or solution error. For an initial grid, the user normally specifies the decimation level.

Templates are used to triangulate the terminated lines quadrilateral mesh as shown in Figure 2. Triangles are

optimized for aspect ratio and the triangle quality is measured by considering the ratio of the area of a triangle to the sum of square of its sides (1). Normalizing this aspect ratio to be 1.0 for an equilateral triangle, the triangulation aims to produce triangles whose aspect ratios are at least 0.5.

Triangles of interior quadrilaterals are of high quality - all triangles are right or acute with high aspect ratios. Further treatment, however, is required for quadrilaterals with region boundaries. As shown in Figure 2b, boundary templates produce triangles of adequate aspect ratio if:

- The boundary does not intersect the quadrilateral very close to its corners.
- The boundary does not contain a vertex point that is very close to a quadrilateral corner or an edge.

The condition of closeness is one third of the quadrilateral side length for producing triangles of aspect ratios greater than 0.5. Quadrilaterals that fail this criteria are preprocessed by a technique called warping (Figure 2c) (2) whereby they are slightly deformed by moving the corner closest to an intersection point onto the intersection point. For vertex points close to a side, a similar concept is used to warp the side onto the vertex point. The resulting polygons are then triangulated producing triangles with aspect ratios greater than 0.5.

Grid Adaptation

Grid adaptation has been achieved for SUPREM-IV diffusion simulation. The SUPREM-IV program has been re-architected such that each of the solvers is run as a client with no static data. At each time step, the grid data from *Forest* is used to create the data areas for diffusion. Upon completion of the time step, the newly computed solution values are updated in *Forest*. Grid adaptation is then performed based on the solution and the diffusion data areas created anew for the next time step.

Adaptation is performed by subdividing the triangular elements based on solution variation (3). Figure 3 shows a simple example of grid adaptation during diffusion simulation. Part of the device is masked during a thermal

anneal in an impurity ambient. As the impurity front diffuses down the device, the grid keeps up with it - finer grid is allocated in areas where the concentration gradient is steepest. For the example presented in Figure 3, grid coarsening was not used.

Interface with String Based Simulators

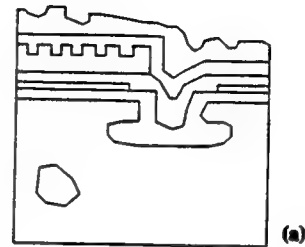
A prototype interface of Forest with the string based deposition simulator SPEEDIE (4) illustrates the power of the quadtree grid generation technique for process simulation. As illustrated in Figure 4, the structure's top layer is extracted from the geometry and a deposition simulation is performed using SPEEDIE. The grid is then subsequently altered to conform to the new boundary. This entails pruning or growth of the quadtree quadrilateral mesh and local triangulation. Minimal changes occur in the existing mesh - thereby reducing interpolation errors between meshes.

User Interface

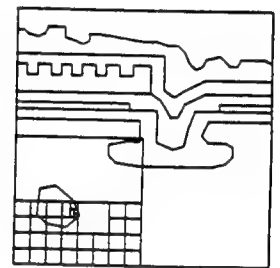
A graphical user interface (GUI) for specifying the initial geometry and initial impurity concentrations has also been developed. The GUI program is a modified version of the Interviews based drawing program *idraw* (5). The program allows the user to graphically draw the device using the mouse. Attributes like material, grid size, doping profiles and boundary conditions are set by mouse operations and by using pull down menus. Text can be used anywhere on the canvas for annotation. The structure is then gridded for device simulation (producing a PISCES mesh file) or for process simulation (producing a SUPREM IV structure file) using Forest. This provides a consistent interface for both types of simulations. Figure 5 shows a typical interface window.

References

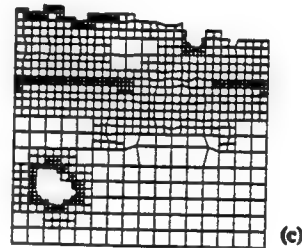
1. R. Bank, "PLTMG User's Guide". SIAM 1990.
2. M. Bern, D. Eppstein and J.R. Gilbert, "Provably good mesh generation." Proc. 31st IEEE Symp. Foundations of Computer Science (1990) 231-241.
3. R. Ismail and G. Amaratunga. "Adaptive meshing schemes for simulating dopant diffusion." IEEE Trans. on CAD. Vol. 9. No 3. March 1990. 276-289.
4. J. McVittie, J. Rey and K. Saraswat, "SPEEDIE User's Manual Version 2.0. Integrated Circuits Laboratory." Stanford University. Feb 1990.
5. J. M. Vlissides. "Generalized graphical object editing". PhD Thesis. Dept. of Elec. Eng. Stanford University. June 1990.



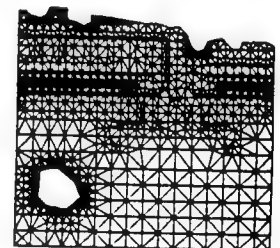
(a)



(b)



(c)



(d)

Figure 1: (a) Quadtree grid generation proceeds by enclosing the geometry by a root square. (b) The root square is decimated recursively. (c) The resulting terminated lines quadrilateral mesh. (d) The final triangulation.

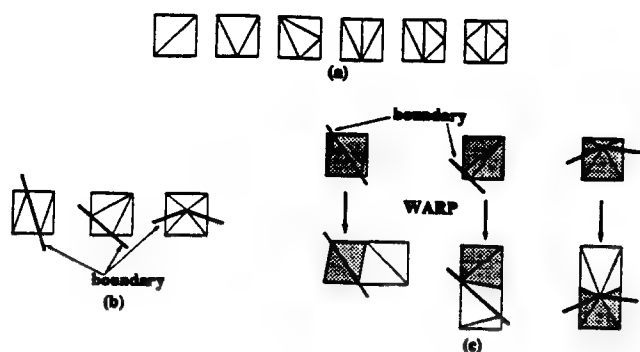


Figure 2: (a) Template used for triangulating interior quadrilaterals. (b) Template used for triangulating boundary quadrilaterals. (c) Warping used to improve triangle qualities of boundary quadrilaterals.

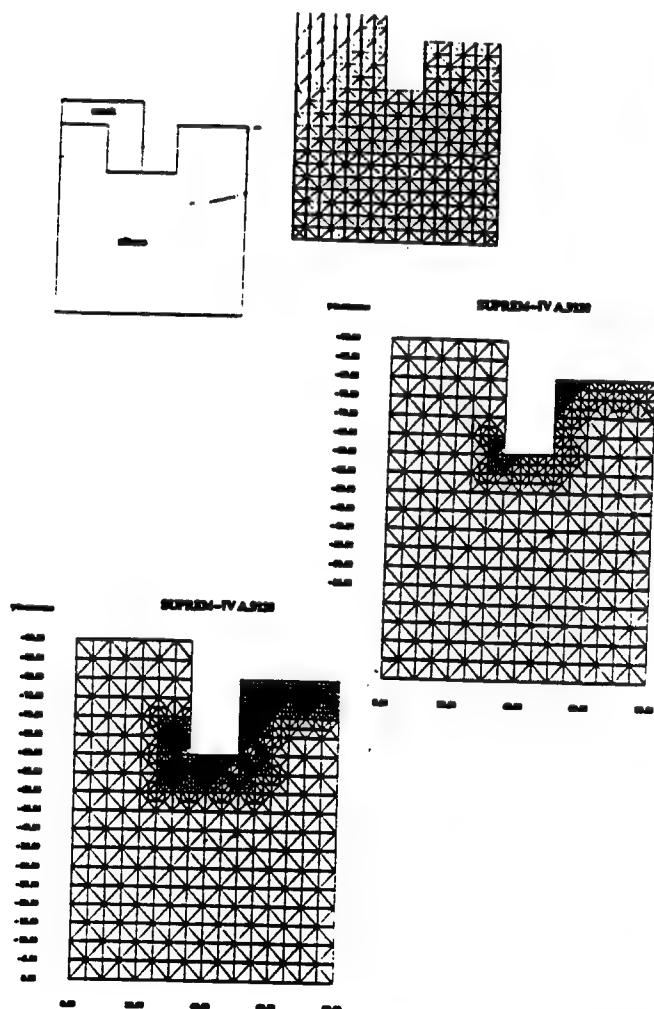


Figure 3: Illustration of grid adaptation during diffusion simulation. The grid is refined at each time step if the solution variation is greater than a threshold.

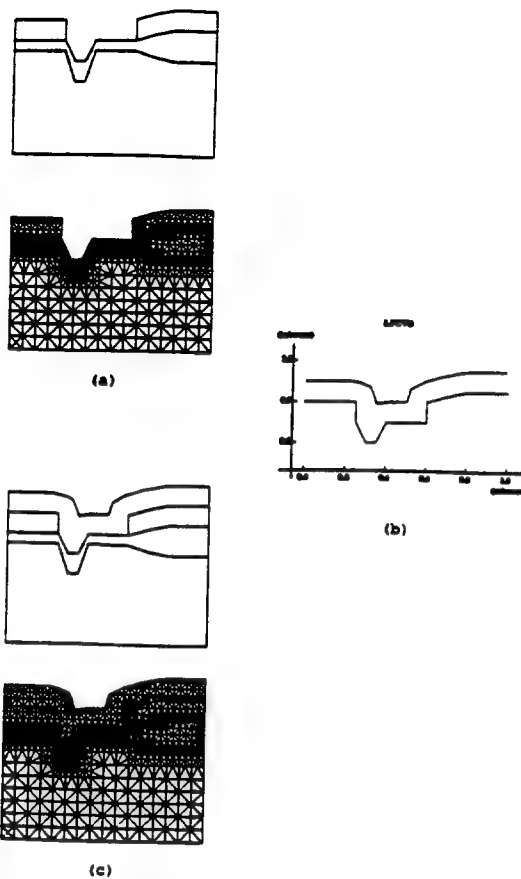


Figure 4: Prototype use of SPEEDIE and Forest for deposition simulation. (a) Grid and Geometry in Forest. (b) Deposition simulation using SPEEDIE. (c) Conforming grid to new boundary.

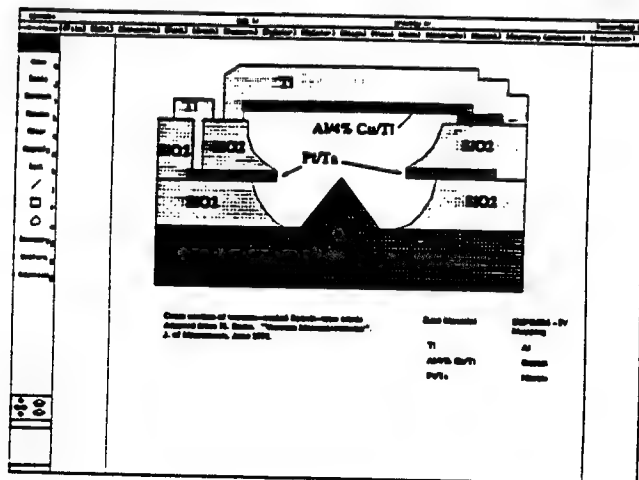


Figure 5: Graphical User Interface for device geometry specification.

An Automated Mesh Refinement Scheme Based On Level-Control Function

Dan Yang

Integrated Circuits Laboratory, Stanford University, Stanford, CA 94305, USA

Kincho Law

Civil Engineering Department, Stanford University, Stanford, CA 94305, USA

Robert Dutton

Integrated Circuits Laboratory, Stanford University, Stanford, CA 94305, USA

Abstract

In order to ensure a good quality mesh, most quadtree/octree based mesh generators abide by the one-level difference rule. This rule requires that each time a rectangle is refined, its neighbors have to be examined. If the one-level difference rule is violated, the neighboring rectangles need to be refined. This refinement propagation can be very expensive in terms of computing time. In this paper, a new refinement algorithm based on the level-control function is proposed for the quadtree mesh generation. This approach eliminates the refinement propagation problem but it requires the level-control function to be defined on the problem domain. For device simulators, the doping profile for initial grids and any error estimate function for adaptive grids are readily good candidates for the level-control functions.

1 Introduction

Quadtree/Octree based mesh generation schemes have been used for finite element methods(FEM) or box methods(BM) to solve PDE problems [1, 3, 4, 8]. There are several advantages for such schemes. First, the basic algorithm is simple and the tree data structure is well suited for most geometry operations (such as finding neighbors, etc.). Second, adaptive mesh refinement can be easily implemented (by dividing a leaf square or pruning a branch of subtree). The scheme can also be integrated nicely with solid modeling since the basic concept was used in computational solid modeling[5]. In the past several years, it has been employed for both 2D and 3D device simulators[4, 6].

A quadtree mesh generation process is a recursive partition of a region of the plane into axis-aligned squares. One square, the root, covers the entire problem region. A square can be divided into four child squares, by splitting it with horizontal and vertical line segments through its center. The collection of squares then forms a tree, with smaller squares at lower levels of the tree[2].

For a mesh suitable for FEM or BM, the method must observe certain requirements. First, we would like the mesh density to be adjustable according to some criteria and also the mesh refinement process should not be too complex. Second a mesh has to be conformal, i.e. no grid point should lie on an edge except at end points. The third requirement relates to the quality of the mesh. A mesh should be absent of obtuse angle elements. It has been well noted that obtuse angle elements can have a severe impact on the simulation process (convergence) as well as simulation results (accuracy).

It is clear that a quadtree/octree mesh meets the mesh density requirement well. The conformity requirement can be met by carrying out the triangulation after a quadtree/octree is created. To satisfy the non-obtuse angle requirement, most quadtree/octree based mesh schemes abide by the one-level difference rule, i.e. for any neighboring leaf squares, the difference of the levels from the root square is at most one. To conform with the one-level difference rule, each time a leaf square is refined, we need to check all neighboring leaf squares and to refine them if necessary. This chain reaction is called refinement propagation. Such a process is inefficient since the same square may be visited many times.

In this paper, we will introduce a new refinement strategy based on the modified level control function. The main idea is that the refinement of any square is based on the values of a level-control function on the four vertices of the square. For example in figure 1, the

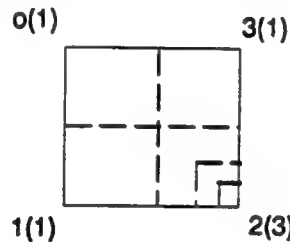


Figure 1: level control-control function

values of level-control function are 1,1, 3 and 1 for vertices 0, 1, 2 and 3 respectively. This implies refining the square one time at vertices 0, 1 and 3 and three times at vertex 2. The result after the refinement is shown as dotted lines.

The rest of the paper is organized as follows. We first discuss the triangulation issue. We then introduce the level-control function, which sets up the foundation for the quantitative analysis of a given quadtree mesh in terms of the mesh density and the mesh quality. Based on the level control function, we present a new refinement scheme which not only generates a suitable mesh satisfying the three requirements but also is optimal with respect to the level-control function. The paper concludes with a brief summary and discussion. Throughout this paper, the discussion is based on 2d meshes. The basic idea, however, can be extended to 3d meshes as well.

2 Quadtree Mesh And Triangulation

To make a quadtree mesh conformal, one easy way is through triangulation. Given a quadtree mesh, there are many ways to carry out triangulation. FEM and BM poses some

basic criteria. One criterion is that any obtuse angle is not desired. However, not all quadtree mesh can be triangulated and still satisfy this condition.

There are some techniques which can be used to overcome this "no-obtuse angle" problem. Let us first introduce a few terms to facilitate our discussion.

- Quad: A square corresponding to a node in the tree.
- Leaf quad: A quad corresponding to the leaf node in the tree.
- Terminal quad: A quad whose children do not make up the quad itself.

A leaf quad is a special terminal quad because it does not have any children.

There are two common ways which can be used to generate a quadtree based conformal mesh. One is by introducing extra vertices during triangulation, which are called Steiner points. For a given quadtree mesh, Steiner points are introduced only in the interior of a terminal quad. The other choice is to force the quadtree to maintain the one-level difference property. A quadtree satisfying one-level difference is also called a balanced tree.

It has been shown that for a balanced quadtree, we can always triangulate it by adding Steiner points in each terminal quad and the number of Steiner points added is bounded by a constant[2].

3 Level Control Function

From the previous discussion, we conclude that if we can construct a balanced quadtree which observes the mesh density requirement, then we will have a conformal mesh with the desired density and without obtuse angle elements.

For a given mesh density, we would like the quadtree to match it as close as possible. We do not want a quadtree to have less grids than desired. On the other hand, too many grids may lead to a waste of the computing resources. In this paper, a concept called level-control function is introduced. The mesh refinement is carried out based on the level-control function. A quad will be further refined if the level of the quad is still less than the level-control function value. It is not difficult to see that for each level-control function, we can generate a quadtree. However, not every level-control function guarantees a balanced quadtree. In other words, not every quadtree based on the level-control function can generate a good mesh. One solution to solve this is to find an approximation for the level-control function such that the approximation will generate a balanced quadtree. There are many ways to find such an approximation. In the following section, we are going to present a scheme which can generate an approximation for any level-control function. The quadtree based on such an approximated level-control function is balanced and we can show that such an approximation is optimal under some measure[7].

4 A New Mesh Refinement Algorithm

In this section, we are going to describe the new refinement algorithm, which has an advantage that the refinement process is localized. Any quad will only be visited once during

the refinement process. Since the quadtree is balanced, it can be triangulated without introducing any new Steiner points.

The refinement is done recursively based on a modified level control function $L'(x)$. $L'(x)$ initially will be equal to $L(x)$. As the refinement progresses, the value of $L'(x)$ will be updated. Let us denote $\text{diff}(v) = |L'(v) - \text{level}(v)|$ in the following discussion. The algorithm is outlined as follows:

1. The refinement process starts from an initial balanced quad-subtree N .
2. For any unrefined quad, we will refine the quad based on the values of $\text{diff}(v)$ where v 's are four vertices of the quad. There are three basic types of refinements, which

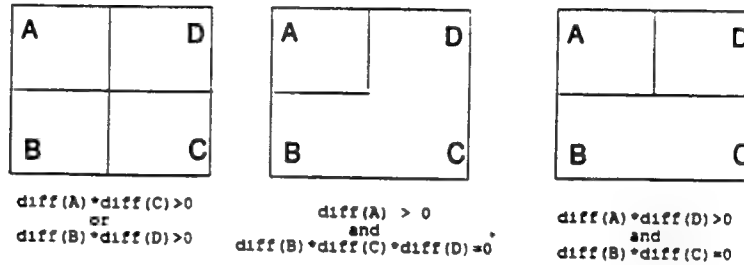


Figure 2: refinement types

are shown in figure 2. All other types are topologically equivalent to those three.

3. After the subdivision, we will update the value for $L'(x)$ on newly added mid-points in the following way. If v is the edge midpoint and assume v_1 and v_2 are two end vertices, then

$$L'(x) = \begin{cases} L'(v) & \text{if } \text{diff}(v_1) * \text{diff}(v_2) > 0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

If v is the center point of the quad, then

$$L'(x) = \begin{cases} L'(v) & \text{if it is a full subdivision} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

4. For each sub-quad, go back to step (2) if more refinement is needed.

It can be shown that the final quadtree is a balanced quadtree and is optimal [7].

Based on the above refinement process, we now present a new mesh generation scheme. This scheme can be used for both initial mesh or mesh refinement. It has three major steps: initialization, refinement and triangulation. For the case of mesh refinement, the initialization step is not needed because the old quadtree can be used as the reference for the initial N .

During the initialization step, we create the initial quadtree N with very few levels. The main purpose of this initial quadtree is to capture those points where level-control function values are high. The next step is the refinement process step which has been discussed earlier. The final step is the triangulation. As noted previously, there are only three types of terminal quads; each can be triangulated without adding any Steiner points, see figure 3. It is clear that diagonals with the opposite orientation will also work.



Figure 3: triangulation

5 Summary And Discussion

Since the refinement is solely determined by the values of the level-control function, each square can be refined independently and no refinement propagation occurs. This indicates that the mesh scheme can possibly be implemented effectively in a parallel computing environment.

One requirement for using this refinement scheme is to have a level-control function defined on the problem domain. In case of generating initial grids for device simulators, the doping profile is a good candidate. The example shown in figure 4 is a bipolar transistor,

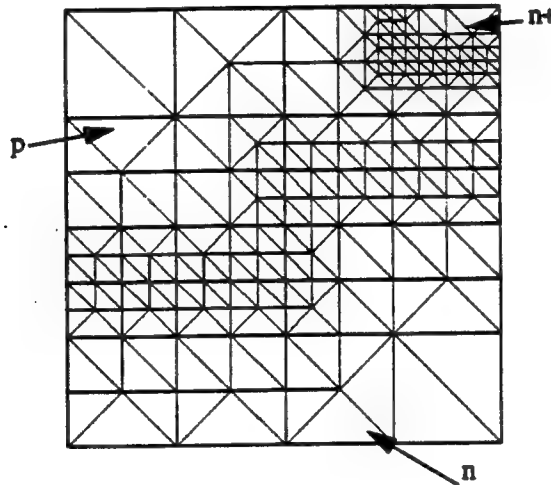


Figure 4: bipolar example

where the level-control function is a modification of the norm of the gradient of the doping profile. In case of adaptive grid, an error estimator function may be used to determine the level-control function.

For device simulators, we often want to have different mesh densities along different directions. This can be achieved by making the level-control function as a vector-function. If this is the case, then the refinement process discussed here has to be modified because it will not guarantee the final quadtree is a balanced tree. In addition, the patterns for terminal triangles in the final quadtree will be more complex and Steiner points may be required.

In summary, a new refinement scheme based on the level-control function is proposed. It

generates a balanced quadtree by given a level control function $L(x)$ and an initial quadtree N . We have also shown that this balanced quadtree is optimal under some measure. The proposed refinement process overcomes the problem of revisiting the same quad many times. The level control function can be easily implemented for device simulators where doping profile and any error estimate function are ready candidates.

6 Acknowledgment

The authors would like to thank Dr. Zhi-Ping Yu for providing the bipolar transistor example.

This work was supported by the State of California under Contract #C90-070 and DARPA Contract #DAAL003-91-C-0043 through the Computing Systems Technology Office.

References

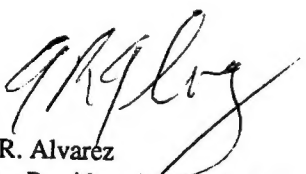
- [1] P. Baehmann, S. Wittchen, M. Shephard, K. Grice and M. Yerry, "Robust, Geometrically Based, Automatic Two Dimensional Mesh Generation," *Int. J. Numer. methods eng.* 24, pp.1043-1078, 1987.
- [2] M. Bern and D. Eppstein, "Mesh Generation And Optimal Triangulation", *Technical Report, Xerox Parc*, 1992.
- [3] F. Cheng and et. al., "A Parallel Mesh Generation Algorithm Based On The Vertex Label Assignment Scheme", *Int. J. Numer. Methods eng.* 28, pp.1429-1448 1989.
- [4] P. Conti, N. Hitschfeld and W. Fichtner, "An Octree-Based Mixed Element Grid Allocator For Adaptive 3D Device Simulation", *NUPAD III Technical Digest* 1990.
- [5] L. Doctor and J. Torborg, "Display Techniques for Octree-Encoded Objects", *IEEE CG&A*, July, pp. 29-37 1981.
- [6] Meshbuild - a 2D mesh generator, Integrated Systems Laboratory, ETH Zurich, Switzerland.
- [7] Dan Yang "Approaches to Initial Gridding" *Technical Report, Stanford Integrated Circuits Lab*,
Int. J. Numer. Methods eng 20, pp. 1965-1990 1984.
- [8] M. Yerry and M. Shephard, "Automatic Three-Dimensional Mesh Generation By The Modified-Octree Technique," *Int. J. Numer. Methods eng.* 20, pp. 1965-1990 1984.



Re: Benefits to Cypress by the Aladdin-CAD Project

Cypress makes over \$100M a year of very fast Static RAMs (SRAMs). It is one of the largest manufacturers of fast Static Rams in the world. The market is such that every 2 to 3 years we must introduce a new generation of SRAMs in order to remain competitive. Every generation must be 4 times the memory capacity, at least as fast, and eventually sell for less than the previous generation. This places a tremendous burden on the development staff to keep up. One of the key enablers to accelerated development cycle times is astute use of Technology Computer-Aided-Design (TCAD). Cypress started supporting the Stanford Aladdin-CAD project approximately 2 years ago for this very purpose. We have felt the interaction to have been extremely positive and rewarding for both sides. In essence, it has exceeded my expectations and we will continue to collaborate on the effort. Below I point out some specifics.

In developing fast SRAM technology one only has to do 3 things: 1) make the world's smallest, most manufacturable cell, 2) make the transistors very fast, and 3) keep the wafers clean. TCAD is instrumental in the first two. From the Aladdin project we were able to obtain a rotatable and expandable three-dimensional (3D) solid model of our full six transistor 0.5 μ memory cell. At 18.4 μ sq this is the smallest six transistor cell ever made. The 3D visualization enabled us to get insights into the cell topology that would have impossible in any other way. If we had had the program available at the beginning of the development effort I believe it would have helped us shave at 3 - 6 months off our schedule and given us a more robust cell. This tool will be used from the beginning in our development of the next generation, sub 10 μ sq cell. In addition to the 3D cell model, our collaboration with Stanford provided us automated routines to generate one-dimensional process simulation impurity profiles. This greatly sped up our ability to generate process options and sensitivity analysis. Finally, we were able help lay the groundwork for full 3D device simulation. While we have not made use of this aspect of the program extensively, for the next generation of technology, 0.35 μ , it will be indispensable.


A.R. Alvarez
Vice-President, Research & Development



International Business Machines Corporation

P.O. Box 100
Somers, NY 10589
914/766-1900

November 15, 1993

Professor Robert W. Dutton
Director of Research
Center for Integrated Systems
Stanford University
Stanford, CA 94305-4070

Dear Dr. Dutton:

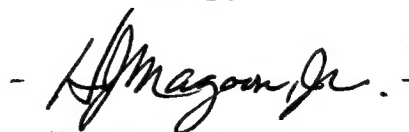
The Aladdin-CAD project has been a unique model of collaboration among competing companies and Stanford University, supported by the State of California office of technology. This project facilitated technology transfer, not only from the university to industry but also from several divisions of IBM to the university. At different phases of the work the IBM Scientific Center, Technical Computing, Research, Development Labs, and Power Parallel Systems have been involved and contributed to the success of the process.

I am very satisfied with the outcomes of this effort. There were natural barriers of competition to be overcome and new areas of cooperation to be explored in this project. Workable solutions to these problems were found. IBM has considered the overall project a success and was able to gain new information and educational benchmarks much more quickly than would otherwise have been possible.

In fact, the success of the Aladdin-CAD project has been influential in IBM's decision to provide \$20 million in Support of University Research in 1993 in the United States. This program will support the development of parallel algorithms, applications and manufacturing methods. Indeed, Stanford University will receive a \$1.3 million grant including a 16 way scalable parallel computer, to continue collaborative research and development to enhance technology transfer, technical job creation in the private sector and US competitive edge.

I look forward to further mutually beneficial collaborative efforts in our present and future endeavors.

Sincerely,



H. J. Magoon, Jr.
Director, PPS Market Support

/vda

Intel Corporation
2200 Mission College Blvd.
P.O. Box 58119
Santa Clara, CA 95052-8119
(408) 765-8080



November 10, 1993

Prof. Robert W. Dutton
Stanford University
Stanford California 94305

Dear Prof. Dutton:

I am writing to give you a summary of the impact of the Aladdin-CAD project on Intel. As you know, several members of Intel's computer modeling group were active participants in this joint project; one group member devoted approximately half of his time for several years to Aladdin-CAD efforts in the development of the 3-D STRIDE device simulator for parallel computers.

At Intel, we are currently using a number of the direct results of the Aladdin-CAD project. The impact of this type of work is greatest in the area of development of next-generation chips, for which the feature sizes must shrink by roughly 30% from previous generation chips. This constant evolution of the technology, leading to faster and cheaper computers, is essential for Intel's survival and continued growth.

Currently, the most widely used product of the Aladdin-CAD project at Intel is STRIDE, the 3-D device simulator. We are using STRIDE for several purposes. First, it is used to calculate parasitic capacitances in newly developed technologies. The ability to accurately and quickly calculate these capacitances has a direct impact on increasing the speed of our new chips. A second important application of STRIDE has been in the design of "substrate taps", which are features of the chip which can only be analyzed using a 3-D simulator. Finally, we have been analyzing 3-D effects in our bipolar transistors using STRIDE. We found that these problems could not have been addressed by using the 2-D simulators more commonly available.

Another important result of the Aladdin-CAD project was the creation of a unified version of PISCES incorporating changes made at Intel and Stanford. This new version greatly aided our effort in bringing Stanford's new energy balance models into Intel. This effort is now bringing us some real benefits, as we have started using the energy balance model to predict the reliability of our next-generation transistors. The ability to accurately predict reliability will allow us to more aggressively increase the speed of our next generation chips.

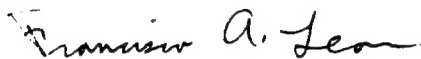
A third area of cooperation that arose from the Aladdin-CAD project was in the use of ACIS for coupling mask layout to solid-modeling. This is an area which has a great potential in the future to automate the analysis of new chip designs and transistor structures.

A fourth area has been Stanford's pioneering work in making the use of parallel computers practical for semiconductor modeling. We are currently making extensive use of the parallel version of BEBOP (from U. of Bologna and Stanford) as well as the parallel version of STRIDE.

Finally, the existence of the formal structure of Aladdin-CAD greatly facilitated a number of informal contacts between Stanford and Intel, the importance of which should not be under-estimated. Stanford researchers have been available for discussion and consultation on a wide variety of topics. The ability for our modeling group to easily interact with Stanford has been a very important advantage for our location in the Bay area.

In summary, while difficult to precisely quantify, the Aladdin-CAD project has had a significant impact on the development of new technology at Intel. I hope that similar joint university/industrial projects will be supported in the future.

Sincerely,

A handwritten signature in cursive script that reads "Francisco A. Leon". The signature is written in dark ink and is positioned to the right of the word "Sincerely,".

Francisco A. Leon
Program Manager, Process and Device Modeling